


Introduction >

General >

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 >

[Java \[Source\]](#)

[Solution](#)

[Lesson Plan](#)

[Hints](#)

[Restart Lesson](#)

## Stage 2

Stage 2: Add Business Layer Access Control.

### **THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT**

Implement a fix to deny unauthorized access to the Delete function. To do this, you will have to alter the WebGoat code. Once you have done this, repeat stage 1 and verify that access to DeleteProfile functionality is properly denied.

**\* You have completed Stage 1: Bypass Business Layer Access Control.**

**\* Welcome to Stage 2: Add Business Layer Access Control**



**Goat Hills Financial**  
Human Resources



Welcome Back [Tom](#) - Staff Listing Page



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 >

Injection Flaws >

Denial of Service >

Insecure Communication >

Insecure Configuration >

stage 3, and verify that access to other employee's profiles is properly denied.

**\* You have completed Stage 3: Bypass Data Layer Access Control.**

**\* Welcome to Stage 4: Add Data Layer Access Control**



## Goat Hills Financial Human Resources

Welcome Back [Tom](#) - View Profile Page

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](#)  
Disciplinary Explanation: [Constantly harassing coworkers](#) Disc. Dates: [10106](#)  
Manager: [102](#)

Access Control Flaws	>
AJAX Security	>
Authentication Flaws	>
Password Strength	
Forgot Password	✔
Multi Level Login 1	✔
Multi Level Login 2	✔
Buffer Overflows	>
Code Quality	>
Concurrency	>
Cross-Site Scripting (XSS)	>
Improper Error Handling	>
Injection Flaws	>
Denial of Service	>
Insecure Communication	>
Insecure Configuration	>

Web applications frequently provide their users the ability to retrieve a forgotten password. Unfortunately, many web applications fail to implement the mechanism properly. The information required to verify the identity of the user is often overly simplistic.

**General Goal(s):**

Users can retrieve their password if they can answer the secret question properly. There is no lock-out mechanism on this 'Forgot Password' page. Your username is 'webgoat' and your favorite color is 'red'. The goal is to retrieve the password of another user.

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

## Webgoat Password Recovery


**For security reasons, please change your password immediately.**

**Results:**

Username: admin

Color: green

Password: 2275\$starBo0rn3

- Access Control Flaws >
- AJAX Security >
- Authentication Flaws >
  - Password Strength
  - Forgot Password 
- Multi Level Login 1
- Multi Level Login 2
- Buffer Overflows >
- Code Quality >
- Concurrency >
- Cross-Site Scripting (XSS) >
- Improper Error Handling >
- Injection Flaws >
- Denial of Service >
- Insecure Communication >

STAGE 2: Now you are a hacker who already has stolen some information from Jane by a phishing mail. You have the password which is tarzan and the Tan #1 which is 15648  
The problem is that the first tan is already used... try to break into the system anyway.

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



## Goat Hills Financial Human Resources



**Firstname:** Jane  
**Lastname:** Plane  
**Credit Card Type:** MC  
**Credit Card Number:** 74589864

[Logout](#)


Access Control Flaws >

AJAX Security >

Authentication Flaws >

Password Strength

Forgot Password 

Multi Level Login 1 

Multi Level Login 2

Buffer Overflows >

Code Quality >

Concurrency >

You are an attacker called Joe. You have a valid account by webgoat financial. Your goal is to log in as Jane. Your username is **Joe** and your password is **banana**. This are your TANS:

Tan #1 = 15161

Tan #2 = 4894

Tan #3 = 18794

Tan #4 = 1564

Tan #5 = 45751

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



**Goat Hills Financial**  
Human Resources



**Firstname:** Jane  
**Lastname:** Plane  
**Credit Card Type:** MC  
**Credit Card Number:** 74589864

[Logout](#)

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	

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:

Search

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

\* You have completed Stage 1: Stored XSS.

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



## Goat Hills Financial Human Resources



Welcome Back [Jerry](#)



First Name:	Tom	Last Name:	Cat
Street:		City/State:	New York, NY
Phone:	443-599-0762	Start Date:	1011999
SSN:	792-14-6364	Salary:	80000
Credit Card:	5481360857968521	Credit Card Limit:	30000
Comments:	Co-Owner.	Manager:	105
Disciplinary Explanation:	NA	Disciplinary Action Dates:	0

ListStaff

EditProfile

DeleteProfile

Logout

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	

Implement a fix to block this reflected XSS attack. Repeat step 5. Verify that the attack is no longer effective.

**\* 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**



- Code Quality >
- Concurrency >
- Cross-Site Scripting (XSS) >
- Phishing with XSS 
- LAB: Cross Site Scripting
- Stage 1: Stored XSS

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

**\* Whoops! You entered for the PIN value 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

Enter your credit card number:

Enter your three digit access code:

Access Control Flaws	>
AJAX Security	>
Authentication Flaws	>
Buffer Overflows	>
Code Quality	>
Concurrency	>
Cross-Site Scripting (XSS)	>
Improper Error Handling	>
Fail Open Authentication Scheme	>
Injection Flaws	>
Denial of Service	>
Insecure Communication	>
Insecure Configuration	>
Insecure Storage	>
Malicious Execution	>
Parameter Tampering	>

Due to an error handling problem in the authentication mechanism, it is possible to authenticate as the 'webgoat' user without entering a password. Try to login as the webgoat user without specifying a password.

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

Welcome, webgoat

You have been authenticated with Fail Open Error Handling

[Logout](#)

[Refresh](#)

## Cookies / Parameters

### Cookies

comment

domain

Burp Intruder Repeater Window Help

Sequencer

Decoder

Comparer

Extender

Project options

User options

Alerts

Target

Proxy

Spider

Scanner

Intruder

Repeater

Intercept

HTTP history

WebSockets history

Options

Request to http://localhost:8080 [127.0.0.1]

Forward

Drop

Intercept is on

Action



Raw

Params

Headers

Hex

```
POST /WebGoat/attack?Screen=114&menu=1100 HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Length: 45
Cookie: JSESSIONID=D6DD7C64CFCCC1B1231CE582C468BCA6
Connection: close
```

Injection Flaws: Command Injection: I tried few times, but still failed to complete this task. sorry for that. Here was what I did.

```
HelpFile=AccessControlMatrix.help" & netstat -ant & ifconfig&SUBMIT=View
```

- Access Control Flaws >
- AJAX Security >
- Authentication Flaws >
- Buffer Overflows >
- Code Quality >
- Concurrency >
- Cross-Site Scripting (XSS) >
- Improper Error Handling >
- Injection Flaws >
  - Command Injection
  - Numeric SQL Injection
  - Log Spoofing
  - XPATH Injection
  - LAB: SQL Injection
    - Stage 1: String SQL Injection
    - Stage 2: Parameterized Query #1

Command injection attacks represent a serious threat to any parameter-driven site. The methods behind an attack are easy to learn and the damage caused can range from considerable to complete system compromise. Despite these risks an incredible number of systems on the internet are susceptible to this form of attack.

Not only is it a threat easily instigated, it is also a threat that, with a little common-sense and forethought, can be almost totally prevented. This lesson will show the student several examples of parameter injection.

It is always good practice to sanitize all input data, especially data that will be used in OS command, scripts, and database queries.

Try to inject a command to the operating system.

You are currently viewing: **AccessControlMatrix.help"**

Select the lesson plan to view:

```
ExecResults for '['/bin/sh, -c, cat "/root/Downloads/.extract/webapps/WebGoat/lesson_plans/en/AccessControlMatrix.html"']'
Returncode: 2
Bad return code (expected 0)
```

Injection Flaws: Command Injection: I tried few times, but still failed to complete this task. sorry for that. Here was what I got.

Log Spoofing

XPATH Injection

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

Modify Data with SQL Injection

Add Data with SQL Injection

Database Backdoors

Blind Numeric SQL Injection

Blind String SQL Injection

Denial of Service >

Insecure Communication >

Insecure Configuration >

in all the weather data being displayed.

**\* 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

AJAX Security >

Authentication Flaws >

Buffer Overflows >

Code Quality >


Concurrency >


Cross-Site Scripting (XSS) >

Improper Error Handling >

Injection Flaws >

Command Injection

Numeric SQL Injection 

Log Spoofing 

XPATH Injection

LAB: SQL Injection

Stage 1: String SQL Injection

Stage 2: Parameterized Query #1

Stage 3: Numeric SQL Injection

- \* 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 attack by adding a script to the log file.

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

User Name :

Password :

Login

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

## Cookies / Parameters

### Cookies

Authentication Flaws	>
Buffer Overflows	>
Code Quality	>
Concurrency	>
Cross-Site Scripting (XSS)	>
Improper Error Handling	>
Injection Flaws	>
Command Injection	
Numeric SQL Injection	✔
Log Spoofing	✔
XPath Injection	✔
LAB: SQL Injection	
Stage 1: String SQL Injection	
Stage 2: Parameterized Query #1	
Stage 3: Numeric SQL Injection	
Stage 4: Parameterized Query #2	

**\* 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

## Injection Flaws >

Command Injection

Numeric SQL Injection 

Log Spoofing 

XPATH Injection 

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 

Modify Data with SQL Injection

Add Data with SQL Injection

Database Backdoors

Blind Numeric SQL Injection

Blind String SQL Injection

command, scripts, and database queries, even if the threat of SQL injection has been prevented in some other manner.

### 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	3334987033333	AMEX		0



## Injection Flaws >

Command Injection

Numeric SQL Injection 

Log Spoofing 

XPATH Injection 

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 

Modify Data with SQL Injection 

Add Data with SQL Injection

Database Backdoors

Blind Numeric SQL Injection

Blind String SQL Injection

Denial of Service >

[Java \[Source\]](#)

[Solution](#)

[Lesson Plan](#)

[Hints](#)

[Restart Lesson](#)

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.

## Cookies / Parameters

### Cookies

comment



## LESSONS

Introduction &gt;

General &gt;

Access Control Flaws &gt;

AJAX Security &gt;

Authentication Flaws &gt;

Buffer Overflows &gt;

Code Quality &gt;

Concurrency &gt;

Cross-Site Scripting (XSS) &gt;

Improper Error Handling &gt;

Injection Flaws &gt;

Command Injection

[Java \[Source\]](#)[Solution](#)[Lesson Plan](#)[Hints](#)[Restart Lesson](#)

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:   No results matched. Try Again.

## Cookies / Parameters



## Blind Numeric SQL Injection



### LESSONS

Introduction >

General >

Access Control Flaws >

AJAX Security >

Authentication Flaws >

Buffer Overflows >

Code Quality >

Concurrency >

Cross-Site Scripting (XSS) >

Improper Error Handling >

Injection Flaws >

Command Injection

[Java \[Source\]](#)

[Solution](#)

[Lesson Plan](#)

[Hints](#)

[Restart 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:



## LESSONS

Introduction >

General >

Access Control Flaws >

AJAX Security >

Authentication Flaws >

Buffer Overflows >

Code Quality >

Concurrency >

Cross-Site Scripting (XSS) >

Improper Error Handling >

Injection Flaws >

Command Injection

[Java \[Source\]](#) [Solution](#) [Lesson Plan](#) [Hints](#) [Restart 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.

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