Nombre Ataque Webgoat	1.Explicación ataque	2.Fácil/difícil	3.Código vulnerable	4.Consecuencias	5.Impacto Bajo/Alto
Num Sql Inj.	Seleccionas el botón Go con cualquier estación predeterminada. De esta manera, al inspeccionar en la red, se puede visualizar el paquete que vamos a reenviar (último paquete attack). Modificamos el cuerpo del paquete e insertamos “station=101 or ‘1’=’1’&SUBMIT=Go! “. A través del “or” y del “1=1”, conseguiremos que el “where” se convierta en true al ser el “or” una unión. La sentencia SQL se convertirá en “SELECT * from weather_data”, mostrando toda la tabla. Por último, enviamos el paquete y vemos en el servidor la respuesta.	El ataque ha sido fácil debido a que con introducir en station la sentencia y reenviar el paquete, era suficientemente para hacer el injection SQL	En el método injectableQuery, se añade directamente al statement la estación. Por lo tanto, no usa PreparedStatement para comprobar que no se añade una SQL Injection ni se valida la entrada de alguna manera	La consecuencia es que se muestra todos los datos de todas las columnas de la tabla weather_data	El impacto es muy alto debido que al hacer una injection sql simple, consigues los datos de toda la tabla
Comm Inj.	Seleccionas el botón View para que, al inspeccionar la red, se pueda visualizar el paquete attack que se va a reenviar. Modificas el cuerpo del paquete insertando “helpfile=”AccessControlMatrix.help;ls”&SUBMIT=VIEW”. De esta manera, se introduce el comando ls dentro del helpfile	El ataque ha sido fácil debido a que solo con introducir el comando, se ejecutará directamente	En el segundo método exec, se pasa como argumento un array de comandos, en el que el tercer argumento es el command injection. Con el método execSimple, se ejecutan todos los comandos, aunque se	Las consecuencias son que puedes introducir comandos que muestren información protegida y que no deba conocer el usuario	El impacto es intermedio dependiendo del comando que puedas introducir, y el tipo de información al que puedas acceder

	para que se ejecute también, y podamos ver los archivos del directorio		suponga que no se “puedan” introducir		
Log Spoof	Seleccionas el botón View para que, al inspeccionar la red, se pueda visualizar el paquete attack que se va a reenviar. Modificas el cuerpo del paquete insertando “username=Smith%0d%0aLogin Succeeded for username: admin<script>alert(document.cookie)</script> &password=guest &SUBMIT=Login. A continuación, reenviamos el paquete. De esta manera, insertamos en el log la url y se muestra al hacer el login que se ha introducido correctamente.	Intermedio. Se considera el ataque intermedio ya que tienes que introducir en formato url el username.	if (inputUsername.length()! = 0). En la anterior comprobación, únicamente comprueba que el username sea de tamaño positivo. Si el tamaño es mayor que 0, procede a validar y decodificar la url.	Las consecuencias son que puedas introducir injections sql en el log del programa, y así descubras o introduzcas nuevos usuarios que no deberían estar.	Alto. Tiene un impacto alto debido a que el log tiene que ser una de las partes más seguras del programa.
Hidden fields	Seleccionamos el botón UpdateCart. Inspeccionamos para ver el paquete attack correspondiente (el último paquete attack). En el cuerpo del paquete, modificamos el price para ponerlo a una cantidad menor correspondiente a la que se debería de pagar (en mi caso 0). Reenviamos el paquete, y se realizaría la compra con el	El ataque ha sido sencillo porque con modificar el atributo precio en el paquete attack, era suficiente para poder pagar por la compra	La validación del precio se comprueba en el comienzo de la clase a través de la variable pattern y script. En cambio, se debería realizar la validación de la entrada en el método createContent, para que se compruebe si es válida la entrada del usuario en el lado del servidor	Las consecuencias son que podrías realizar una compra que costase una gran cantidad de dinero, y que únicamente tengas que pagar una cantidad de dinero irrisoria por ella.	El impacto es muy alto, porque en este caso, una compra de altas cantidades podría salir gratis. Con lo cual, la empresa acarrearía enormes pérdidas y problemas.

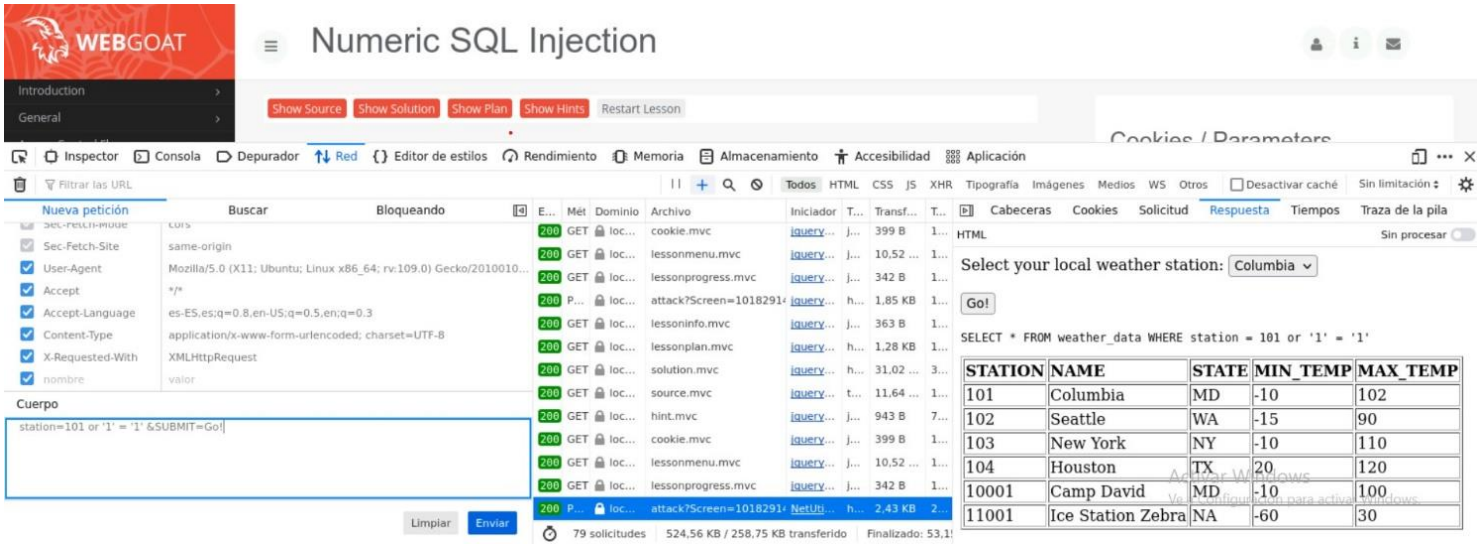
	precio que se haya establecido anteriormente.				
Buffer Overflow	Seleccionamos el botón submit para visualizar el modelo del paquete attack que vamos a reenviar. En segundo lugar, copiamos más de 4096 caracteres en el parámetro 3 (utilizamos el carácter A), que en este caso es el Room Number. Modificamos dicho argumento en el cuerpo del paquete "first_name=&last_name=&room_no=A……..A", y reenviamos el paquete. A continuación, seleccionamos el nuevo paquete attack que se envía y aceptamos los términos en la siguiente pantalla. Seleccionamos el nuevo paquete attack, y en respuesta, seleccionamos sin procesar. Se podrá visualizar todos los datos en "value" ordenados. Basta con volver a la página principal con los datos de uno de los usuarios para poder introducirte en el sistema del hotel	El ataque ha sido difícil porque se requiere primero en analizar si es posible un buffer overflow, y en ese caso, a partir de cuantos valores. Además, se necesita visualizar y modificar varios paquetes	En el método makeThirdStep, se encuentra la siguiente comprobación "if(param3.length() > 4096)". Si se cumple la condición, se añaden todos los datos de los usuarios. Por lo tanto, se podrá utilizar cada uno de los datos para registrarse	Las consecuencias son que se puede registrar con cada uno de los usuarios, pudiendo hacer cualquier tipo de acción en la página del hotel	El impacto es muy alto ya que con los datos de los usuarios se podrá acceder a cualquier parte del hotel, y se podría cancelar o realizar reservas, gastar dinero…
Thread Safety	Seleccionamos la tarea Thread Safety Problems. Abrimos otra misma pestaña copiando la url.	El ataque ha sido muy fácil porque con abrir una	En el método createContent, se realiza una comprobación para ver	Las consecuencias son que, entrando con tu registro, consigues la información de	El impacto es muy alto debido a que con la información de registro

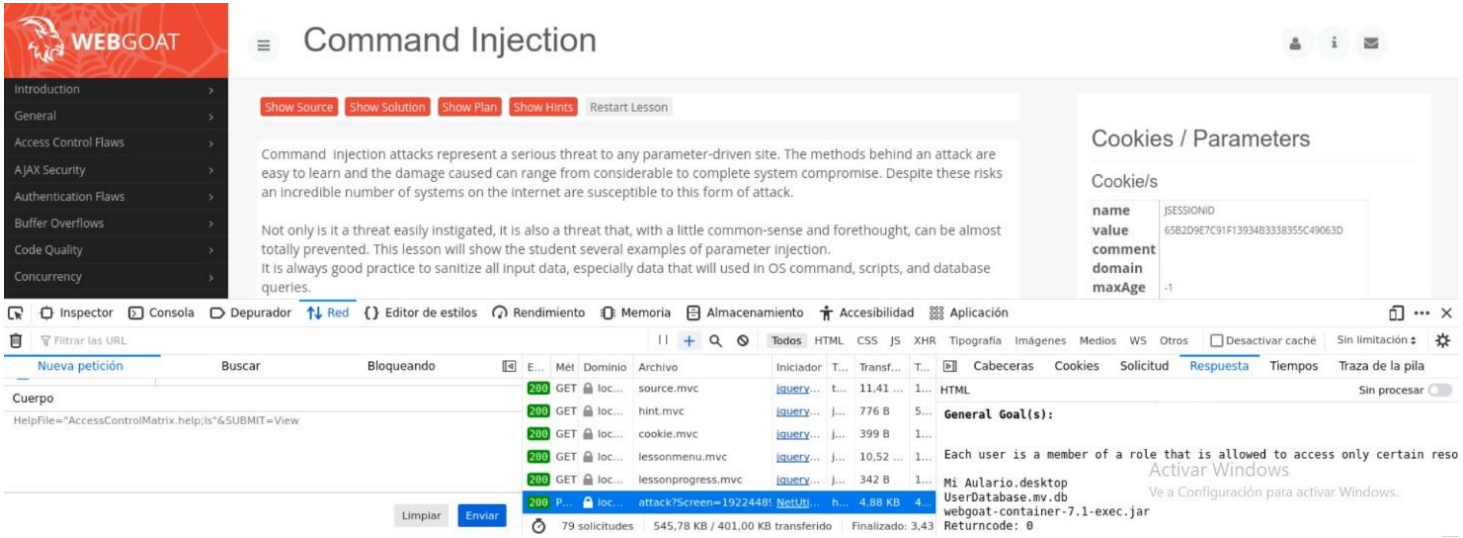
	Como tenemos dos usuarios (jeff y daves), introducimos un nombre en cada pestaña. Seleccionamos el botón submit a la vez en ambas pestañas, y se mostrará la información del registro de uno de los dos usuarios en las dos pestañas (la información del segundo usuario).	nueva pestaña e introducir los dos usuarios, era suficiente para conseguir la información de registro del otro usuario	si se está con el usuario del momento. El problema está en que antes de seleccionar la información de la base de datos, se realiza un sleep de 1.5 segundos. En ese tiempo, otro usuario podría intentar acceder a su respectiva información de registro. Con lo cual, el nuevo usuario será el usuario del momento, y el antiguo usuario al terminar el sleep obtendrá la información del nuevo usuario.	registro de otro usuario con el que podrás acceder de ahora en adelante	se podrá realizar cualquier tipo de acción con riesgo en la aplicación
ZipBomb	Creamos un archivo a través de Python que ocupe más de 20 mb. Para ello, simplemente escribimos múltiples espacios o caracteres en el archivo txt. Comprimimos el zip, y lo añadimos a la tarea para subirlo	El ataque ha sido fácil porque con crear un zip de determinado tamaño (máximo 2mb), y con cuyos archivos de su interior ocupen más de 20 mb, era necesario para realizar el ataque	En el método handleRequest, se realizan dos comprobaciones con los tamaños de 20 y 2 mb. En primer lugar, se valida que el archivo ocupe menos de 2mb (“if (item.getSize() < 2000 * 1024”). A continuación, si los archivos del interior ocupan más de 20 mb (“if (total > 20 * 1024 * 1024”), se añaden a la webSesion y se crean mensajes	Las consecuencias son que puedes saturar o hacer que el programa se quede sin memoria porque no está preparado para una entrada de tal tamaño. Esto podría bloquear la aplicación y que no se pudiera utilizar ante tal carga de memoria	El impacto es alto debido a que con un introducir un zip en la aplicación, podrías conseguir que se sature toda la aplicación, y que los usuarios no pudieran utilizarla
Malicious Execution	Creamos un archivo jsp e introducimos un código	El ataque ha sido fácil porque se	String uploaded_file_path =uploads_and_target_pare	Las consecuencias son que puedas ejecutar cualquier	El impacto es alto debido a que puedes ejecutar código que tenga

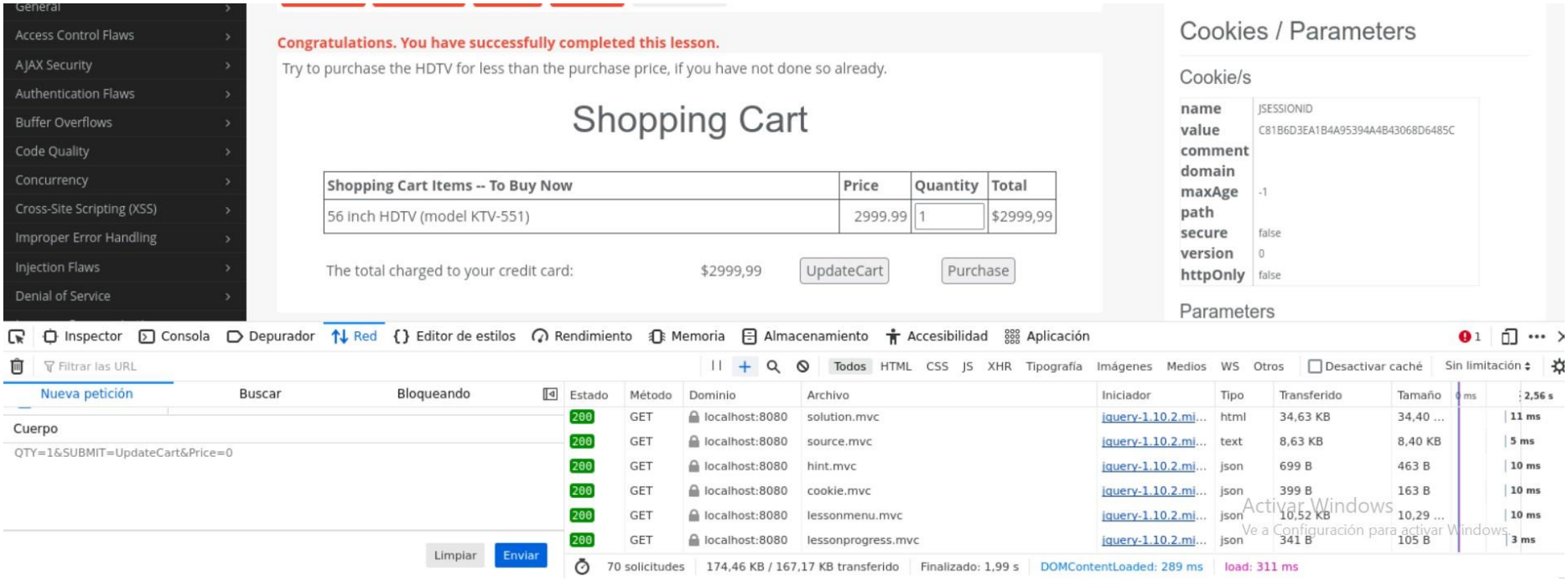
	cualquiera para que se ejecute. En mi caso, he creado el archivo jsp en netbeans. He introducido en el interior el código que proporciona webgoat para crear el file en el directorio establecido. A continuación, seleccionamos el botón browse y el archivo jsp. Para finalizar, seleccionamos Start Upload para subir el archivo	podía introducir un código cualquiera dentro del archivo jsp, y con subir el archivo era suficiente para realizar el ataque	nt_directory + UPLOADS_RELATIVE_PATH + java.io.File.separator + item.getName(); File uploadedFile = new File(uploaded_file_path); En el siguiente código, se puede comprobar cómo no se verifica si la extensión es segura. De esta manera, puedes introducir un archivo malicioso.	tipo de código peligroso dentro del archivo jsp. En el ejemplo, se crea un file llamado guest.txt en el directorio establecido dentro del Escritorio.	mucho riesgo, y el grado de posibilidad de ataques es enorme porque depende en su totalidad del código que introduzcas

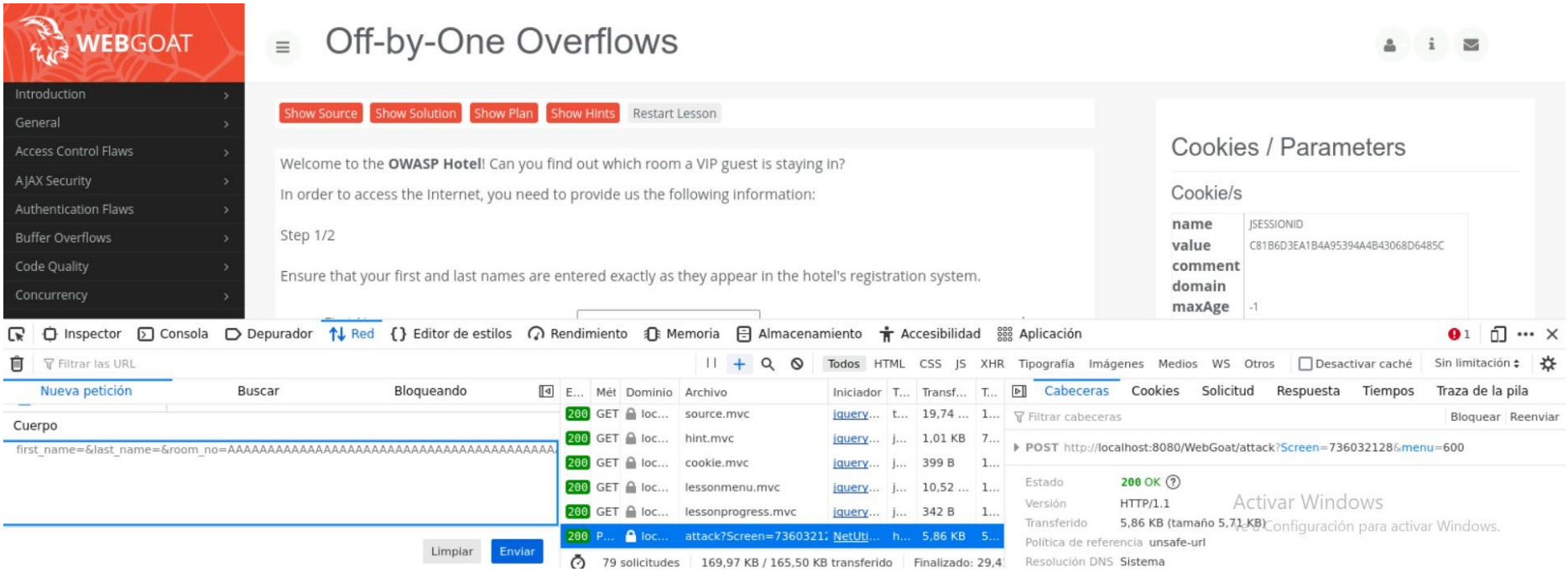
Nombre Ataque Webgoat	6.CWE correspondiente	7.Ataque CWE dificultad contraria	8.Consecuencia CWE contrarias
Num Sql Inj.	- CWE-20: Improper Input Validation (no se valida la entrada)	Un posible ataque complicado sería que en el código hubiera mucha validación de entrada y se tuviera que analizar cada una de las partes para ir poco a poco descubriendo vulnerabilidades	Las posibles consecuencias contrarias serían que la tabla mostrada fuera de nula utilidad al conocer con anterioridad todos esos datos
Comm Inj.	- CWE-89: Improper Neutralization of Special Elements used in an-SQL Command ('SQL Injection')	Un posible ataque complicado sería en el caso de ser comprobados la mayoría de los comandos que no se pueden introducir, y haga falta ir comprobando cuáles no están siendo prevenidos	Las posibles consecuencias contrarias serían que se introdujera un comando de baja utilidad que no proporcione ningún riesgo
Log Spoof	CWE-117: Improper Output Neutralization for Logs	Un posible ataque más simple sería que no hiciera falta ni codificar el username a url y que únicamente se compruebe que el tamaño sea positivo	Las posibles consecuencias contrarias serían que no se pueda dañar los archivos logs
Hidden fields	- CWE-89: Improper Neutralization of Special Elements used in an-SQL Command ('SQL Injection')	Un posible ataque más complicado sería solo se pudiera modificar el precio de algunos productos, y se tuviera que ir analizando en cuales de los productos se puede bajar el precio.	Las posibles consecuencias contrarias serían que no se pudieran prácticamente modificar los precios de los artículos, o solo algunos de precio y cantidad pequeña.

Buffer Overflow	- CWE-120: Classic Buffer Overflow	Un posible ataque más simple sería el no tener que estar visualizando varios paquetes attack, debido a que con modificar uno solo de ellos es suficiente.	Las posibles consecuencias contrarias serían que, en vez de conseguir los datos de los usuarios, se pudieran conseguir datos mucho menos irrelevantes para el sistema
Thread Safety	- CWE-307: Improper Restriction of Excessive Authentication Attempts	Un posible ataque más complicado sería que hicieran falta crear múltiples hilos y no solo dos como en este caso, y que al hacer ello, se requiriera de mucha memoria	Las posibles consecuencias contrarias serían que se obtuviera un tipo de información carente de valor en vez de la información de registro de un usuario
ZipBomb	-CWE-434: Unrestricted Upload of File with Dangerous Type	Un posible ataque más complicado sería que se restringiera el tamaño mucho más, o que se permitiera una entrada de muchísimo tamaño haciendo que sea muy difícil crear un archivo de tanto tamaño	Las posibles consecuencias contrarias serían que el tamaño del zip que se introdujera no consiguiera saturaciones haciendo que tenga un riesgo mucho menor
Malicious Execution	-CWE-434: Unrestricted Upload of File with Dangerous Type	Un posible ataque más complicado sería que validara la entrada, pero de forma incorrecta, dejando algún fallo en algún lado. En ese caso, se debería ver qué tipo de ataque no está siendo comprobado	Las posibles consecuencias contrarias serían que el código introducido no tuviera prácticamente ningún tipo de vulnerabilidad y no pusiera en riesgo a la aplicación


WEBGOAT
Numeric SQL Injection
Introduction
General
Show Source
Show Solution
Show Plan
Show Hints
Restart Lesson
Cookies / Parameters
Inspector
Consola
Depurador
Red
Editor de estilos
Rendimiento
Memoria
Almacenamiento
Accesibilidad
Aplicación
Nueva petición
Buscar
Bloqueando
Sec-Fetch-Site
same-origin
User-Agent
Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/2010010...
Accept
/
Accept-Language
es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Content-Type
application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With
XMLHttpRequest
Cuerpo
station=101 or '1' = '1' &SUBMIT=Go!
Limpiar
Enviar
79 solicitudes
524,56 KB / 258,75 KB transferido
Finalizado: 53,1
Cabeceras
Cookies
Solicitud
Respuesta
Tiempos
Traza de la pila
Sin procesar
Select your local weather station:
Columbia
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


WEBGOAT
Command Injection
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Show Source
Show Solution
Show Plan
Show Hints
Restart Lesson
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 used in OS command, scripts, and database queries.
Cookies / Parameters
Cookie/s
name JSESSIONID
value 65B2D9E7C91F13934B3338355C49063D
comment
domain
maxAge -1
Inspector
Consola
Depurador
Red
Editor de estilos
Rendimiento
Memoria
Almacenamiento
Accesibilidad
Aplicación
Nueva petición
Buscar
Bloqueando
Cuerpo
HelpFile="AccessControlMatrix.help;ls"&SUBMIT=View
Limpiar
Enviar
79 solicitudes
545,78 KB / 401,00 KB transferido
Finalizado: 3,43
Cabeceras
Cookies
Solicitud
Respuesta
Tiempos
Traza de la pila
Sin procesar
General Goal(s):
Each user is a member of a role that is allowed to access only certain reso
Mi Aulario.desktop
UserDatabase.mv.db
webgoat-container-7.1-exec.jar
Returncode: 0


General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Congratulations. You have successfully completed this lesson.
Try to purchase the HDTV for less than the purchase price, if you have not done so already.
Shopping Cart
Shopping Cart Items -- To Buy Now
Price
Quantity
Total
56 inch HDTV (model KTV-551)
2999.99
1
$2999,99
The total charged to your credit card:
$2999,99
UpdateCart
Purchase
Cookies / Parameters
Cookie/s
name JSESSIONID
value C81B6D3EA1B4A95394A4B43068D6485C
comment
domain
maxAge -1
path
secure false
version 0
httpOnly false
Parameters
Inspector
Consola
Depurador
Red
Editor de estilos
Rendimiento
Memoria
Almacenamiento
Accesibilidad
Aplicación
Filtrar las URL
Todos HTML CSS JS XHR Tipografía Imágenes Medios WS Otros
Desactivar caché
Sin limitación
Nueva petición
Buscar
Bloqueando
Cuerpo
QTY=1&SUBMIT=UpdateCart&Price=0
Limpiar
Enviar
Estado Método Dominio Archivo Iniciador Tipo Transferido Tamaño
200 GET localhost:8080 solution.mvc jquery-1.10.2.mi... html 34,63 KB 34,40 ... 11 ms
200 GET localhost:8080 source.mvc jquery-1.10.2.mi... text 8,63 KB 8,40 KB 5 ms
200 GET localhost:8080 hint.mvc jquery-1.10.2.mi... json 699 B 463 B 10 ms
200 GET localhost:8080 cookie.mvc jquery-1.10.2.mi... json 399 B 163 B 10 ms
200 GET localhost:8080 lessonmenu.mvc jquery-1.10.2.mi... json 10,52 KB 10,29 ... 10 ms
200 GET localhost:8080 lessonprogress.mvc jquery-1.10.2.mi... json 341 B 105 B 3 ms
70 solicitudes
174,46 KB / 167,17 KB transferido
Finalizado: 1,99 s
DOMContentLoaded: 289 ms
load: 311 ms
Activar Windows
Ve a Configuración para activar Windows.

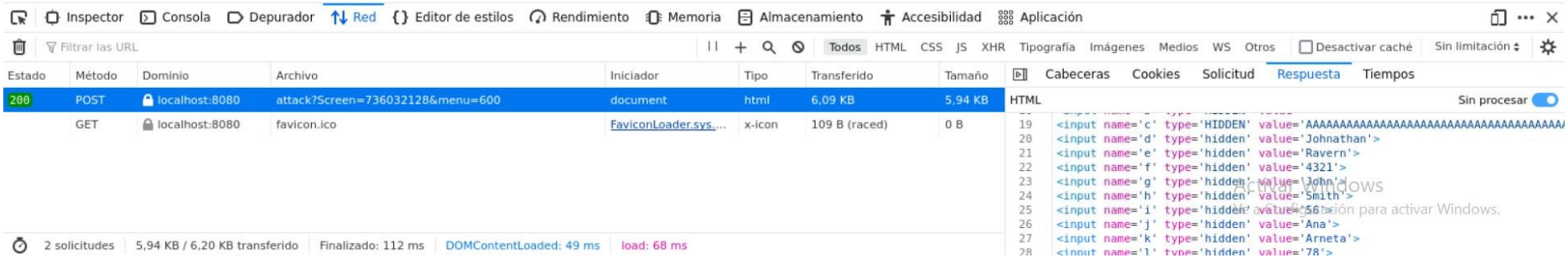

WEBGOAT
Off-by-One Overflows
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Show Source
Show Solution
Show Plan
Show Hints
Restart Lesson
Welcome to the OWASP Hotel! Can you find out which room a VIP guest is staying in?
In order to access the Internet, you need to provide us the following information:
Step 1/2
Ensure that your first and last names are entered exactly as they appear in the hotel's registration system.
Cookies / Parameters
Cookie/s
name JSESSIONID
value C81B6D3EA1B4A95394A4B43068D6485C
comment
domain
maxAge -1
Inspector
Consola
Depurador
Red
Editor de estilos
Rendimiento
Memoria
Almacenamiento
Accesibilidad
Aplicación
Filtrar las URL
Todos HTML CSS JS XHR Tipografía Imágenes Medios WS Otros
Desactivar caché
Sin limitación
Nueva petición
Buscar
Bloqueando
Cuerpo
first_name=&last_name=&room_no=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Limpiar
Enviar
E... Mét Dominio Archivo Iniciador T... Transf... T...
200 GET loc... source.mvc jquery... t... 19,74 ... 1...
200 GET loc... hint.mvc jquery... j... 1,01 KB 7...
200 GET loc... cookie.mvc jquery... j... 399 B 1...
200 GET loc... lessonmenu.mvc jquery... j... 10,52 ... 1...
200 GET loc... lessonprogress.mvc jquery... j... 342 B 1...
200 P... loc... attack?Screen=7360321 NetUti... h... 5,86 KB 5...
79 solicitudes 169,97 KB / 165,50 KB transferido Finalizado: 29,4
Cabeceras Cookies Solicitud Respuesta Tiempos Traza de la pila
Filtrar cabeceras
Bloquear Reenviar
POST http://localhost:8080/WebGoat/attack?Screen=736032128&menu=600
Estado 200 OK
Versión HTTP/1.1
Transferido 5,86 KB (tamaño 5,71 KB)
Política de referencia unsafe-url
Resolución DNS Sistema
Activar Windows
Ve a Configuración para activar Windows.

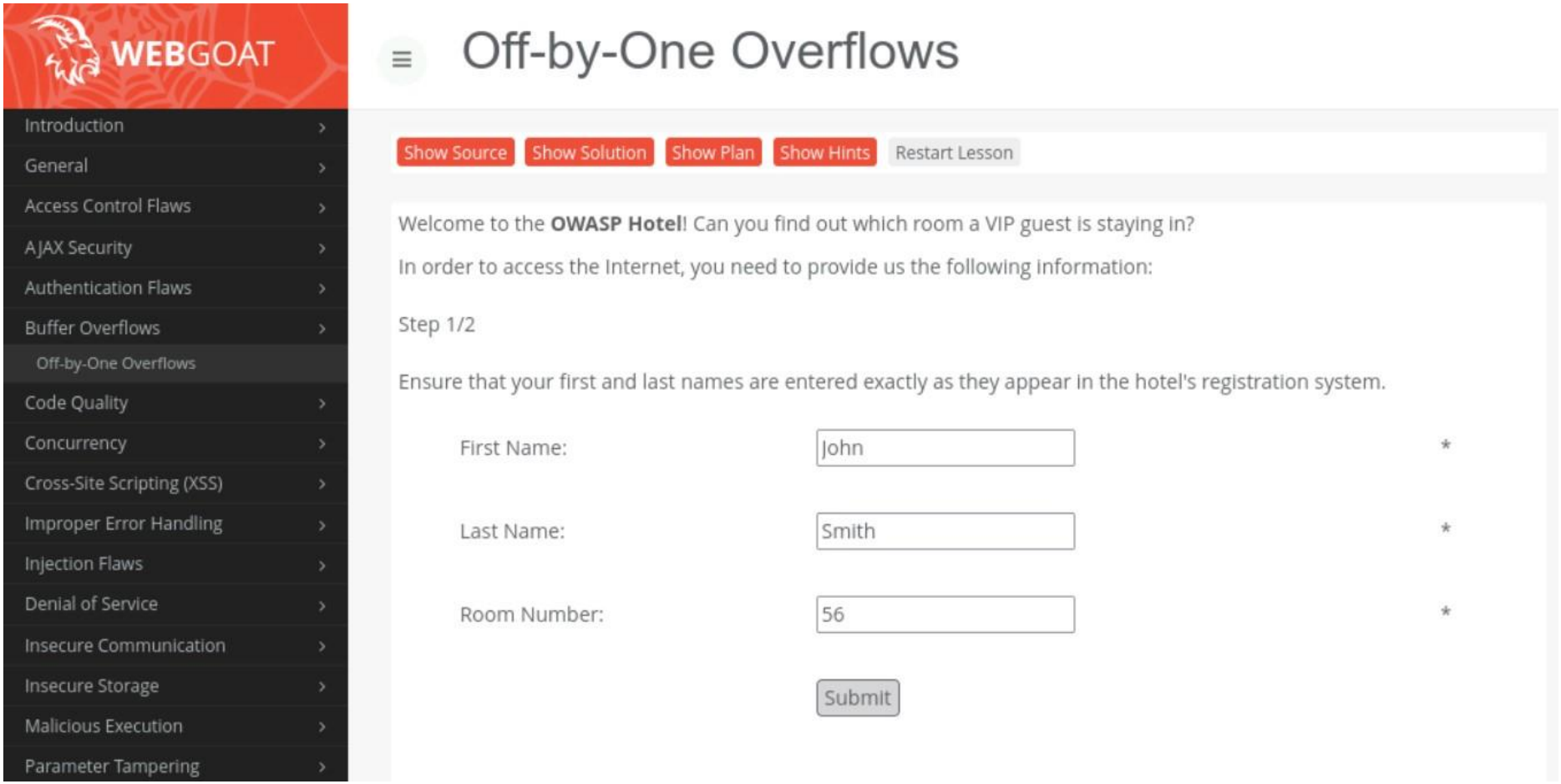
Welcome to the **OWASP Hotel**! Can you find out which room a VIP guest is staying in?

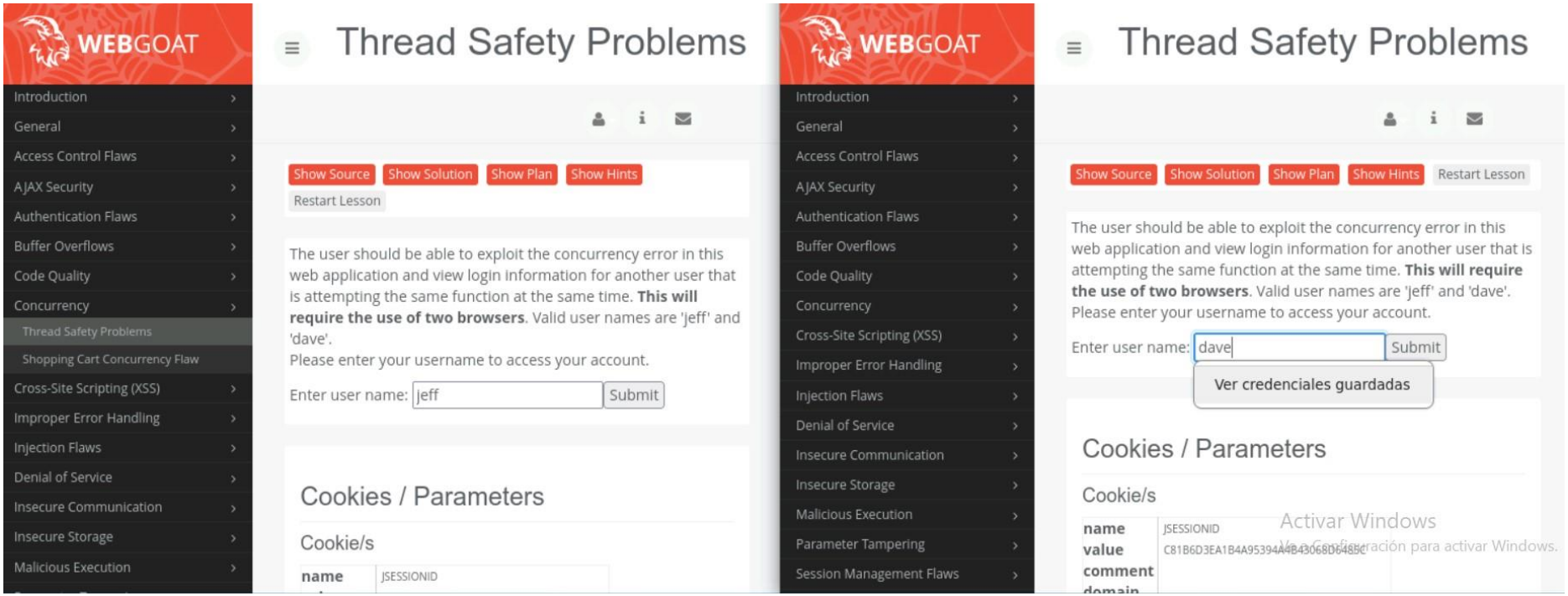
* To complete the lesson, restart lesson and enter VIP first/last name
You have now completed the 2 step process and have access to the Internet

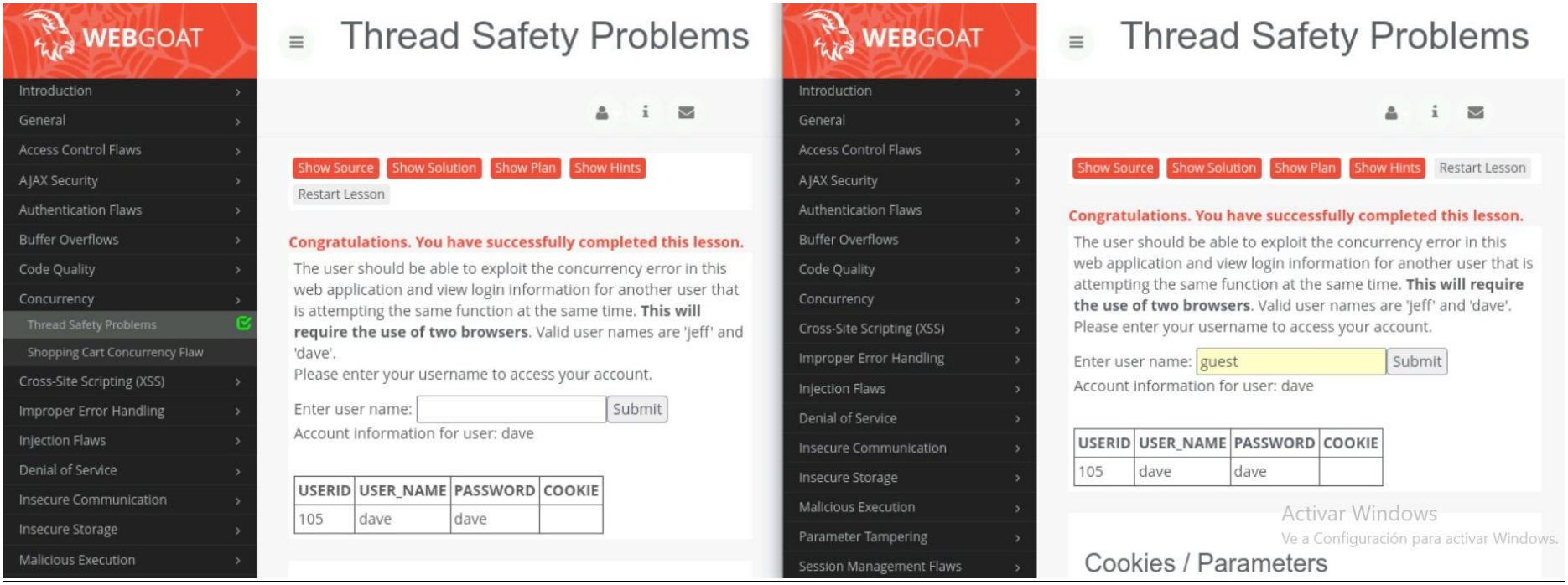
Process complete

Your connection will remain active for the time allocated for starting now.

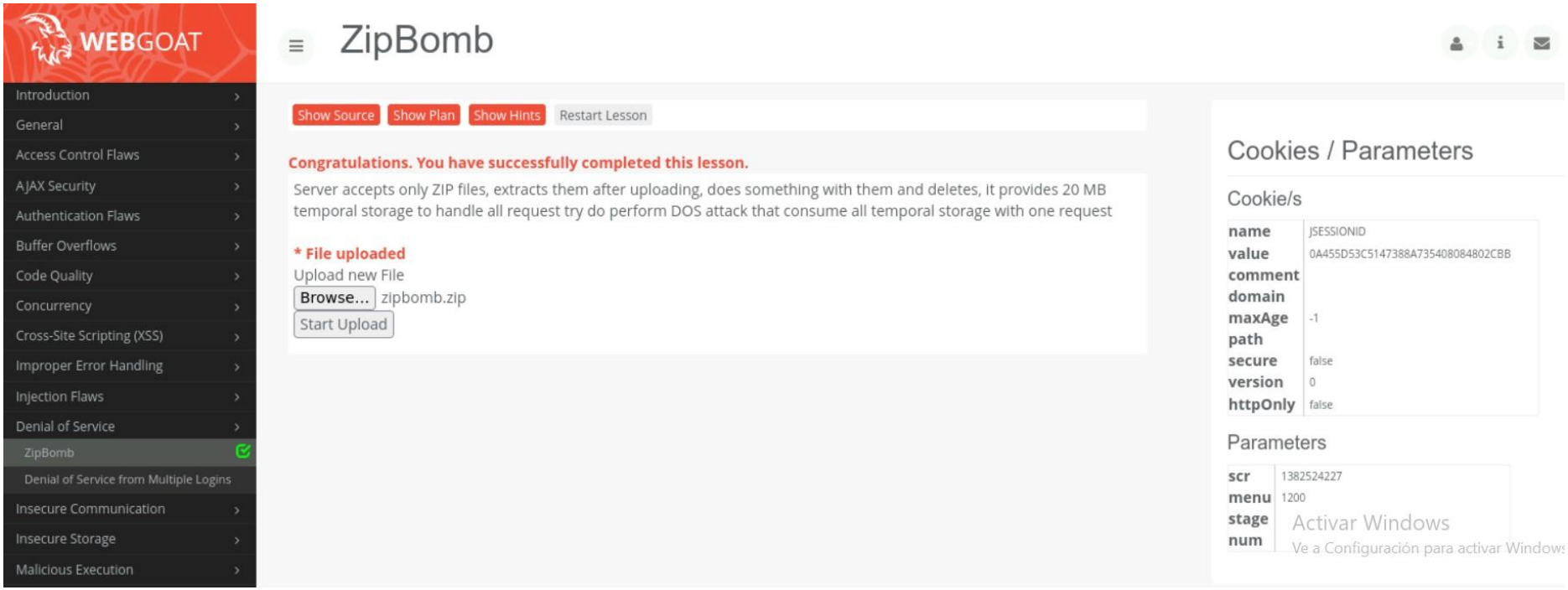



WEBGOAT
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Off-by-One Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Storage
Malicious Execution
Parameter Tampering
Off-by-One Overflows
Show Source
Show Solution
Show Plan
Show Hints
Restart Lesson
Welcome to the OWASP Hotel! Can you find out which room a VIP guest is staying in?
In order to access the Internet, you need to provide us the following information:
Step 1/2
Ensure that your first and last names are entered exactly as they appear in the hotel's registration system.
First Name:
John
*
Last Name:
Smith
*
Room Number:
56
*
Submit


WEBGOAT
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Thread Safety Problems
Shopping Cart Concurrency Flaw
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Storage
Malicious Execution
Thread Safety Problems
Show Source
Show Solution
Show Plan
Show Hints
Restart Lesson
The user should be able to exploit the concurrency error in this web application and view login information for another user that is attempting the same function at the same time. This will require the use of two browsers. Valid user names are 'jeff' and 'dave'.
Please enter your username to access your account.
Enter user name: jeff
Submit
Cookies / Parameters
Cookie/s
name
JSESSIONID
WEBGOAT
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Storage
Malicious Execution
Parameter Tampering
Session Management Flaws
Thread Safety Problems
Show Source
Show Solution
Show Plan
Show Hints
Restart Lesson
The user should be able to exploit the concurrency error in this web application and view login information for another user that is attempting the same function at the same time. This will require the use of two browsers. Valid user names are 'jeff' and 'dave'.
Please enter your username to access your account.
Enter user name: dave
Submit
Ver credenciales guardadas
Cookies / Parameters
Cookie/s
name
value
comment
JSESSIONID
C81B6D3EA1B4A95394A4B43068D6485C
Activar Windows
Ve a Configuración para activar Windows.



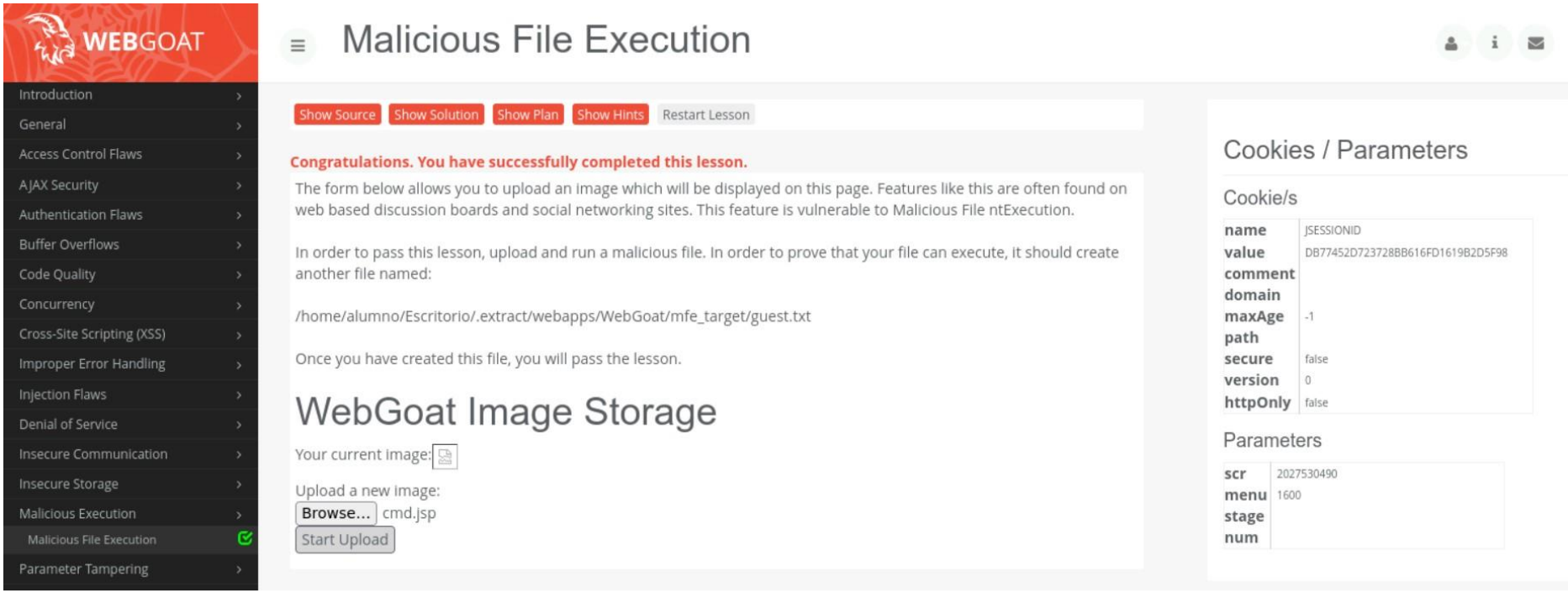

```
f = open("zipbomb.txt","w");
txt = " "*10**5
for i in range(500):
  f.write(txt)
  print(i)
f.close()
```


WEBGOAT
ZipBomb
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
ZipBomb
Denial of Service from Multiple Logins
Insecure Communication
Insecure Storage
Malicious Execution
Show Source
Show Plan
Show Hints
Restart Lesson
Congratulations. You have successfully completed this lesson.
Server accepts only ZIP files, extracts them after uploading, does something with them and deletes, it provides 20 MB temporal storage to handle all request try do perform DOS attack that consume all temporal storage with one request
* File uploaded
Upload new File
Browse...
zipbomb.zip
Start Upload
Cookies / Parameters
Cookie/s
name JSESSIONID
value 0A455D53C5147388A735408084802CBB
comment
domain
maxAge -1
path
secure false
version 0
httpOnly false
Parameters
scr 1382524227
menu 1200
stage
num
Activar Windows
Ve a Configuración para activar Windows

cmd: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<HTML>
<%
java.io.File file = new java.io.File("/home/alumno/Escritorio/.extract/webapps/WebGoat/mfe_target/guest.txt");
file.createNewFile();
%>
</HTML>
```


WEBGOAT
Malicious File Execution
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Storage
Malicious Execution
Malicious File Execution
Parameter Tampering
Show Source
Show Solution
Show Plan
Show Hints
Restart Lesson
Congratulations. You have successfully completed this lesson.
The form below allows you to upload an image which will be displayed on this page. Features like this are often found on web based discussion boards and social networking sites. This feature is vulnerable to Malicious File ntExecution.
In order to pass this lesson, upload and run a malicious file. In order to prove that your file can execute, it should create another file named:
/home/alumno/Escritorio/.extract/webapps/WebGoat/mfe_target/guest.txt
Once you have created this file, you will pass the lesson.
WebGoat Image Storage
Your current image:
Upload a new image:
Browse...
cmd.jsp
Start Upload
Cookies / Parameters
Cookie/s
name JSESSIONID
value DB77452D723728BB616FD1619B2D5F98
comment
domain
maxAge -1
path
secure false
version 0
httpOnly false
Parameters
scr 2027530490
menu 1600
stage
num