



# Security Lab



Student Name:

# Lab 2: Information Gathering and WebGoat Attacks Lab

This lab uses the **Kali 2021** virtual machine (VM) as OVA file **KALI-20.ova** on Canvas. The credentials are as follows:

Username: osboxes

Password: osboxes.org

## Information Gathering

---

Before attacking (i.e., pen testing) a website, we need to gather some important value to map the attack surface. If we don't understand how the site is working, what is available on the site, what type of input it takes etc., then we will not be able to launch a good attack.

Gathering information and mapping the site is very important, so you will be given a basic understanding of a few tools that can be used but you will need to dig deeper to attack a system.

## Lab Tasks

**nslookup** (Name Server Lookup) is the name of a program that lets an Internet server administrator, or any computer user enter a host name. You may access nslookup using the Terminal Emulator.

**If you encounter any issues while using the update command use the following three commands first and proceed with update command(Step 1)**

**# download**

```
wget http://http.kali.org/kali/pool/main/k/kali-archive-keyring/kali-archive-keyring_2022.1_all.deb
```

**# install**

```
sudo dpkg -i kali-archive-keyring_2022.1_all.deb
```

**# remove downloaded file again**

```
rm kali-archive-keyring_2022.1_all.deb
```

**Step 1:** Only if nslookup is not installed:

- `sudo apt-get update`
- `sudo apt-get install dnsutils`

`nslookup` followed by the domain name will display the "A Record" (IP Address) of the domain.

Use this command to find the address record for `google.com`:

```
> nslookup google.com
```

**Q1: Attach a screenshot of your results from this `nslookup` command.**

We can also view all the available DNS records using the `-type=any` option.

```
> nslookup -type=any google.com
```

**Q2: Attach a screenshot of your results from this `nslookup` command. Since there can be a lot of information presented here, only include the actual data between the server/address (found in the earlier query) and the authoritative answers (where there should be several authoritative answers that start with `google.com`).**

**Q3: Does the information help you as an attacker? If yes, how? Otherwise, if no, why?**

**nmap** (Network Mapper) is a security scanner used to discover hosts and services on a computer network, thus creating a "map" of the network. To accomplish its goal, `nmap` sends specially crafted packets to the target host and then analyzes the responses.

The software provides several features for probing computer networks, including host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features. `nmap` is also capable of adapting to network conditions including latency and congestion during a scan.

Only if `nmap` is not installed:

- `sudo apt-get install nmap`

Now, let's find our gateway's open ports as well as OS name and version using `nmap`. Note that this command requires root privileges.

```
> sudo nmap -O -v 129.120.210.235
```

**Q4: Attach a screenshot of your results from this `nmap` command.**

## Q5: Does the information help you as an attacker? If yes, how? Otherwise, if no, why?

### References:

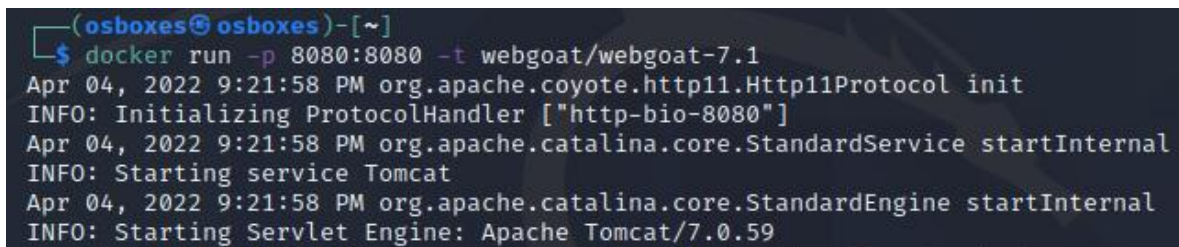
- 1) <http://www.debianhelp.co.uk/nslookup.htm>
- 2) [http://nmap.org/nmap\\_doc.html](http://nmap.org/nmap_doc.html)

## Webgoat

---

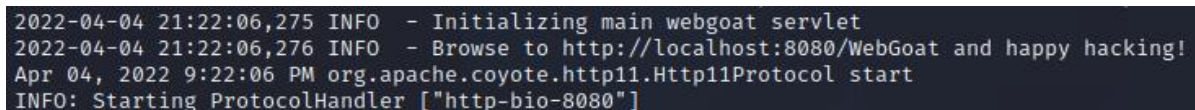
1. Type the `docker` command to start the web server as shown below:

```
> docker run -p 8080:8080 -t webgoat/webgoat-7.1
```



```
(osboxes@osboxes)-[~]  
$ docker run -p 8080:8080 -t webgoat/webgoat-7.1  
Apr 04, 2022 9:21:58 PM org.apache.coyote.http11.Http11Protocol init  
INFO: Initializing ProtocolHandler ["http-bio-8080"]  
Apr 04, 2022 9:21:58 PM org.apache.catalina.core.StandardService startInternal  
INFO: Starting service Tomcat  
Apr 04, 2022 9:21:58 PM org.apache.catalina.core.StandardEngine startInternal  
INFO: Starting Servlet Engine: Apache Tomcat/7.0.59
```

2. You will then get a web server started (no additional command here):

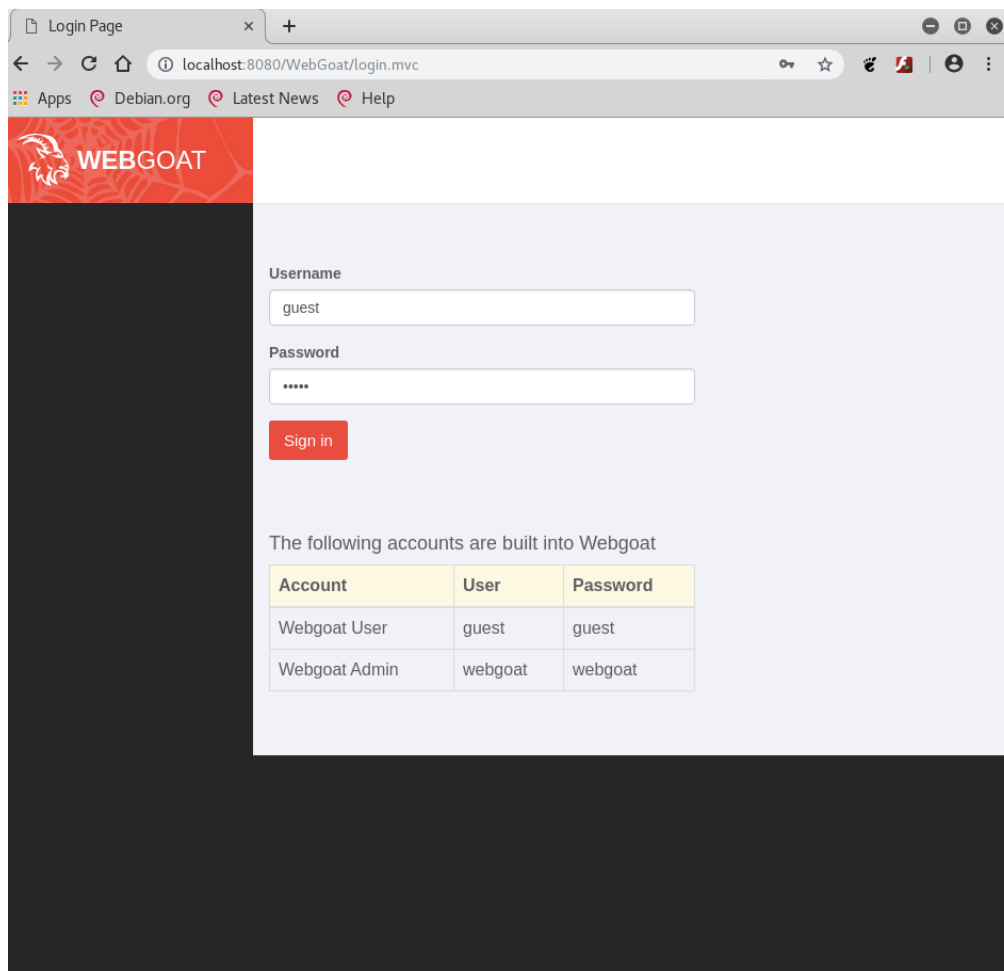


```
2022-04-04 21:22:06,275 INFO - Initializing main webgoat servlet  
2022-04-04 21:22:06,276 INFO - Browse to http://localhost:8080/WebGoat and happy hacking!  
Apr 04, 2022 9:22:06 PM org.apache.coyote.http11.Http11Protocol start  
INFO: Starting ProtocolHandler ["http-bio-8080"]  
█
```

Normally, you should see a message as above if WebGoat's Apache Tomcat server started successfully. Occasionally, the notifications as above are not shown, so please proceed with the next command, regardless of whether you see the notifications or not.

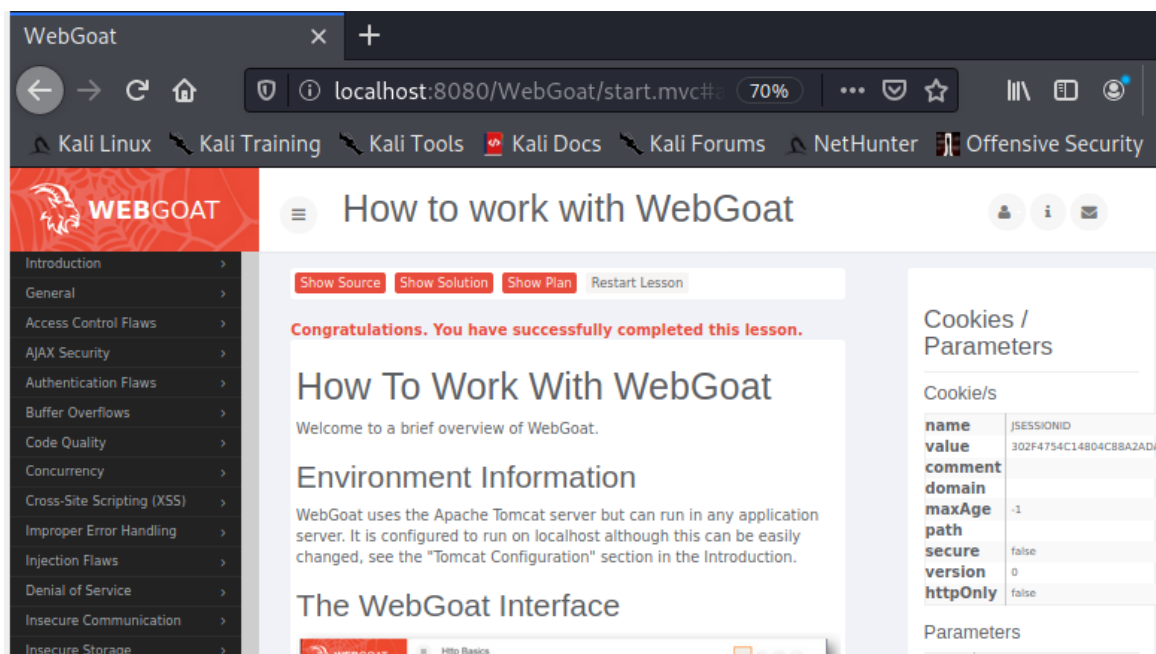
Type: `"localhost:8080/WebGoat/login.mvc"` in the browser address bar to start WebGoat.

Default passwords are shown on the below screenshot:



Account	User	Password
Webgoat User	guest	guest
Webgoat Admin	webgoat	webgoat

Use the "guest" login to start WebGoat.



Congratulations. You have successfully completed this lesson.

## How To Work With WebGoat

Welcome to a brief overview of WebGoat.

### Environment Information

WebGoat uses the Apache Tomcat server but can run in any application server. It is configured to run on localhost although this can be easily changed, see the "Tomcat Configuration" section in the Introduction.

### The WebGoat Interface

#### Cookies / Parameters

##### Cookie/s

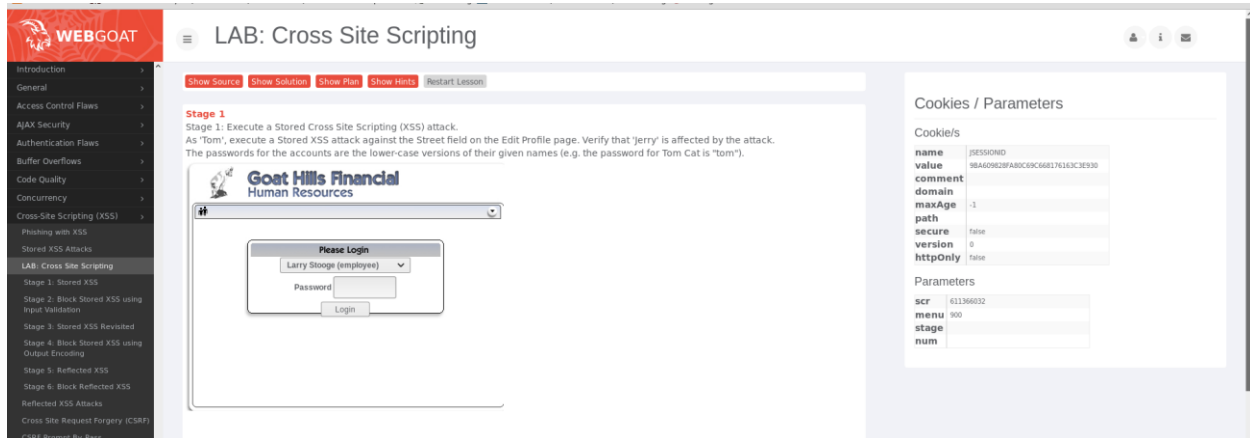
name	JSESSIONID
value	302F4754C14804CB8A2ADA
comment	
domain	
maxAge	-1
path	
secure	false
version	0
httpOnly	false

##### Parameters

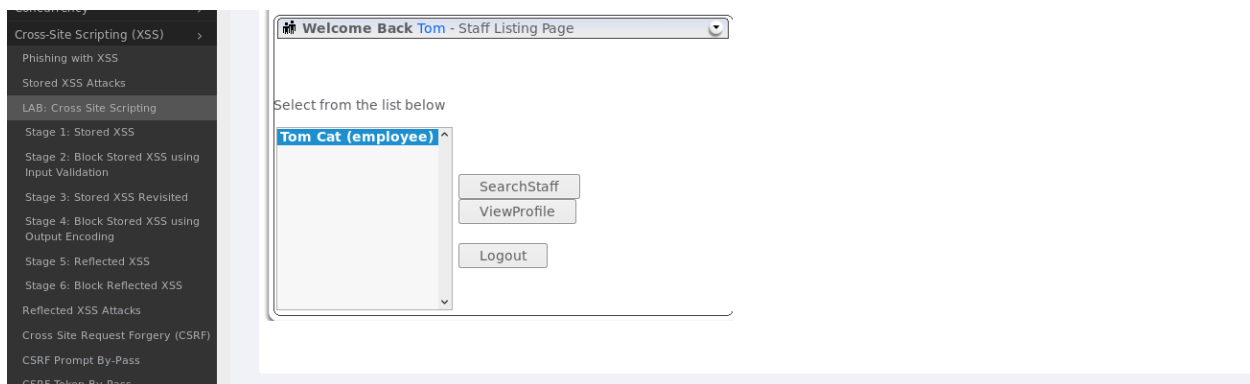
# CROSS SITE SCRIPTING

Select the following option in the WebGoat left hand menu:

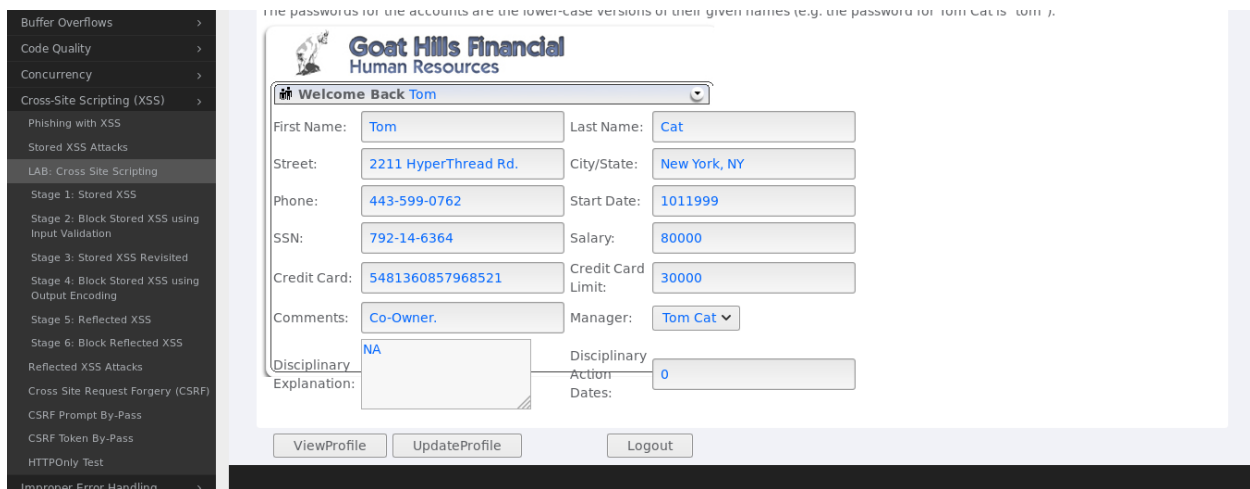
**Cross Site Scripting → LAB: Cross Site Scripting**



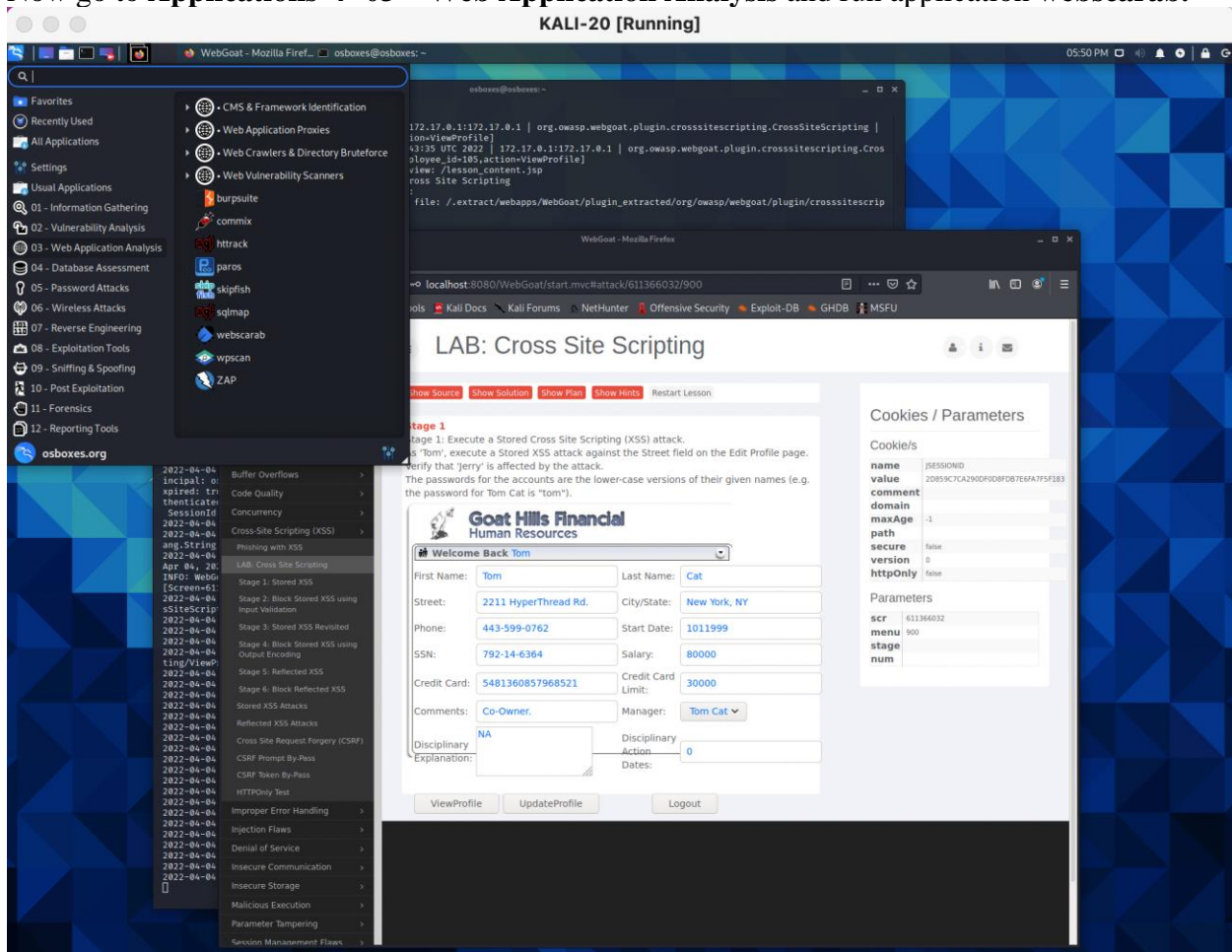
**Login** as user Tom Cat (employee) where password is "tom" in the above page. You should then see the following page or similar.



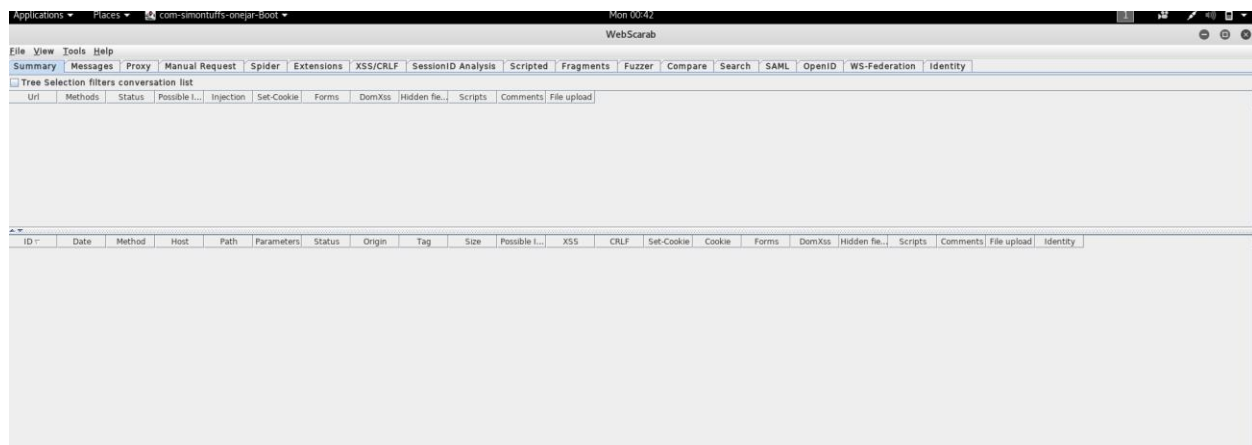
Select Tom Cat (employee) and then click on **View Profile**. Now click on **Edit Profile** to see the following screen.



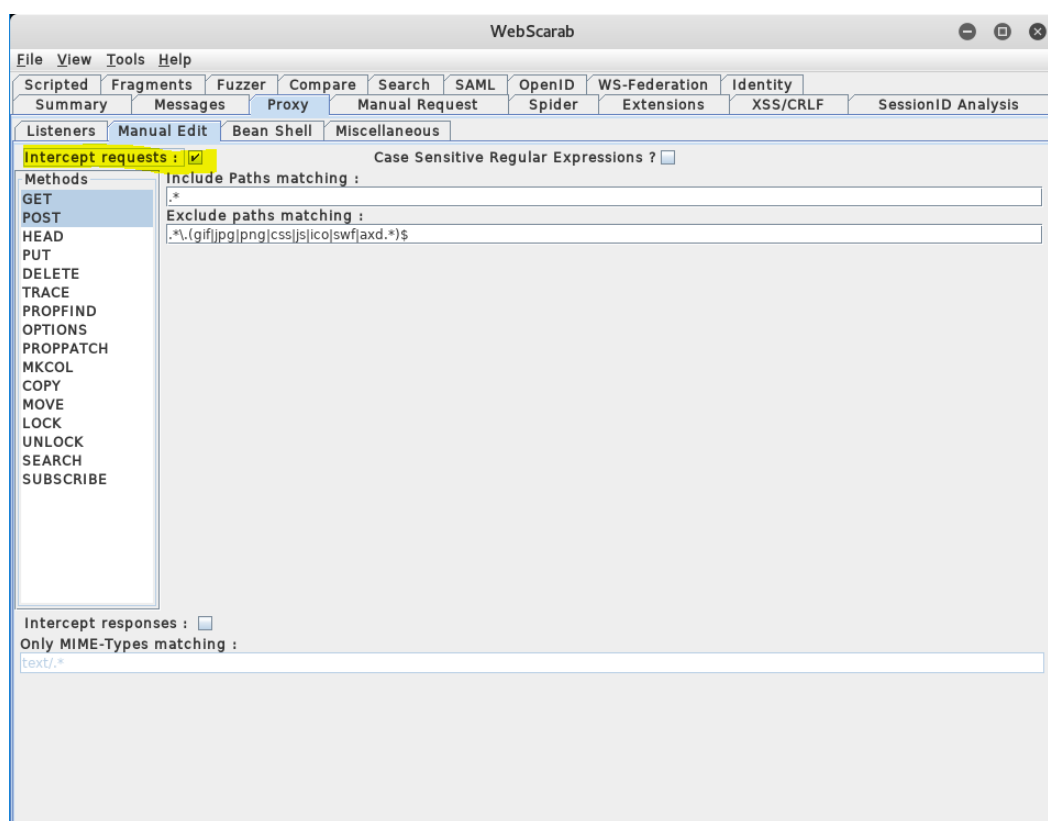
Now go to **Applications → 03 – Web Application Analysis** and run application **webscarab**.



This application is used to manipulate the data being sent using the HTTP "POST" method. After opening **webscarab**, you should see the following screen:

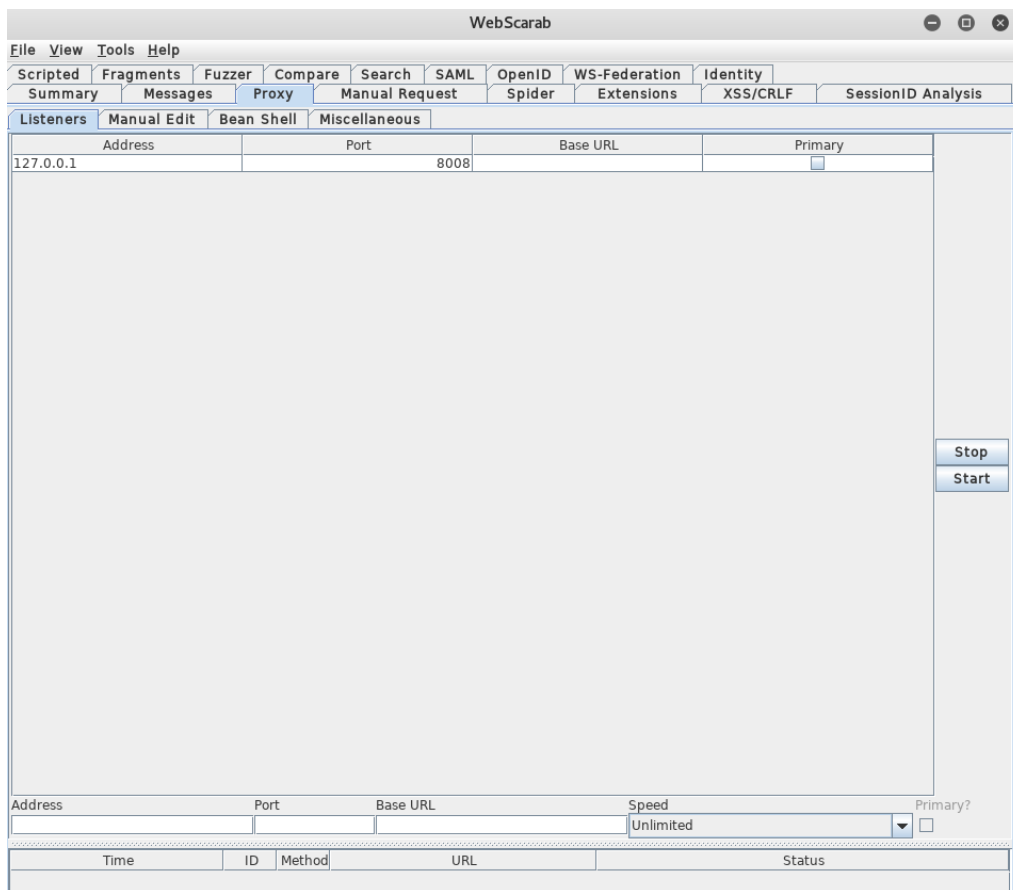


Go to **Proxy** → **Manual Edit** and select **Intercept requests**.

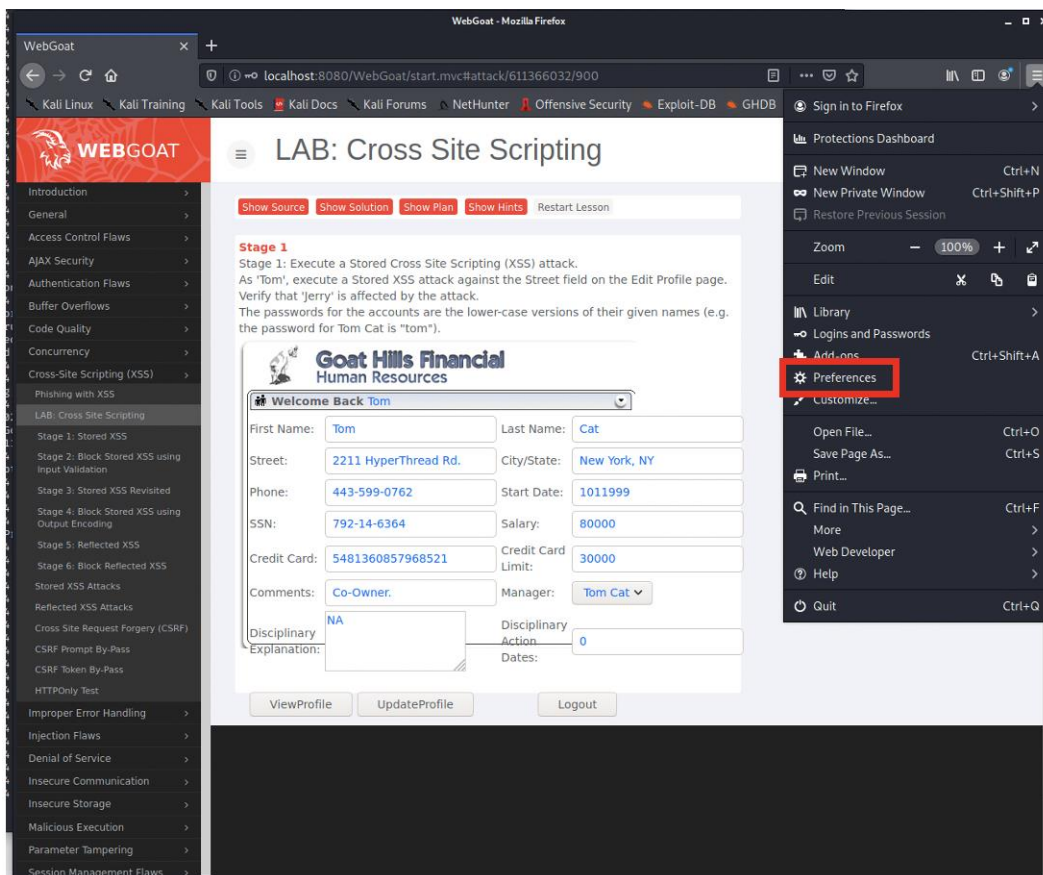


Go to **Proxy** → **Listeners** and observe the port and address (in the case shown, it is 127.0.0.1 and 8008).

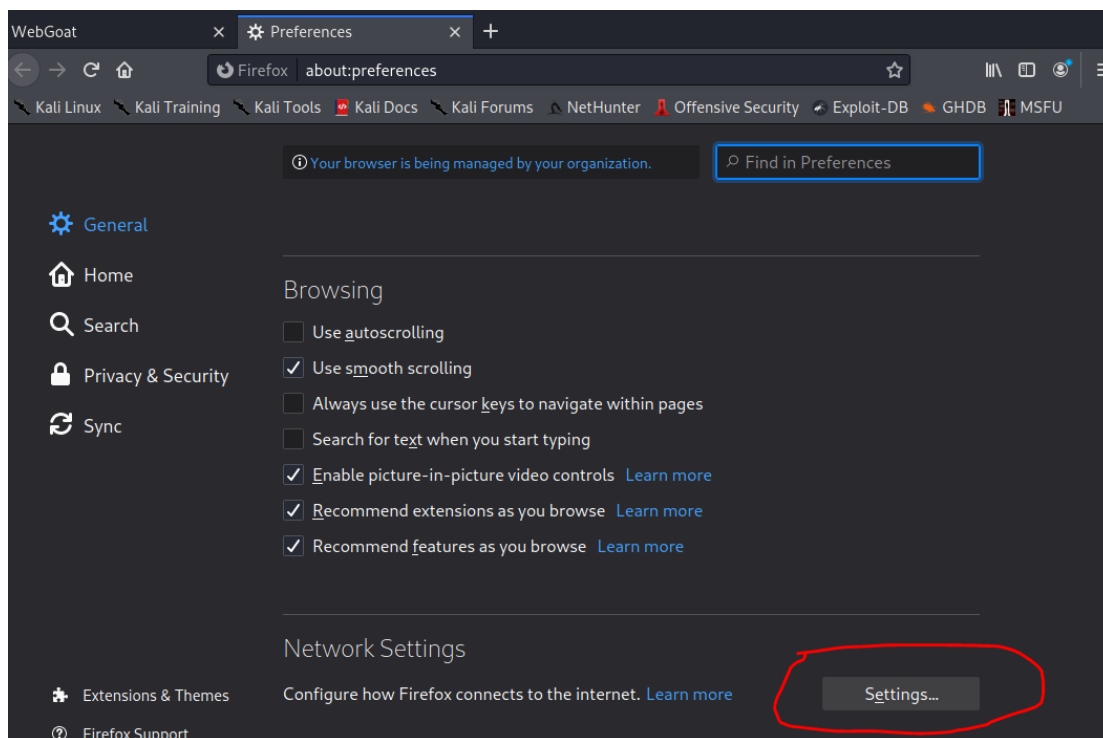




Now go to your browser (for WebGoat) and open **Preferences** in the menu.



Under **General**, scroll down to find **Network Settings** and then click on **Settings**.



Enter the settings as below and, if applicable, remove any text in **No proxy for** box. Enter the address (127.0.0.1) and port number (8008) from **webscarab** and click **OK**.

Connection Settings

**Configure Proxy Access to the Internet**

☐ No proxy

☐ Auto-detect proxy settings for this network

☐ Use system proxy settings

☒ Manual proxy configuration

HTTP Proxy 127.0.0.1 Port 8008

☒ Also use this proxy for FTP and HTTPS

HTTPS Proxy 127.0.0.1 Port 8008

FTP Proxy 127.0.0.1 Port 8008

SOCKS Host 127.0.0.1 Port 8008

☐ SOCKS v4 ☒ SOCKS v5

☐ Automatic proxy configuration URL

Reload

No proxy for

Example: .mozilla.org, .net.nz, 192.168.1.0/24

Connections to localhost, 127.0.0.1, and ::1 are never proxied.

☐ Do not prompt for authentication if password is saved

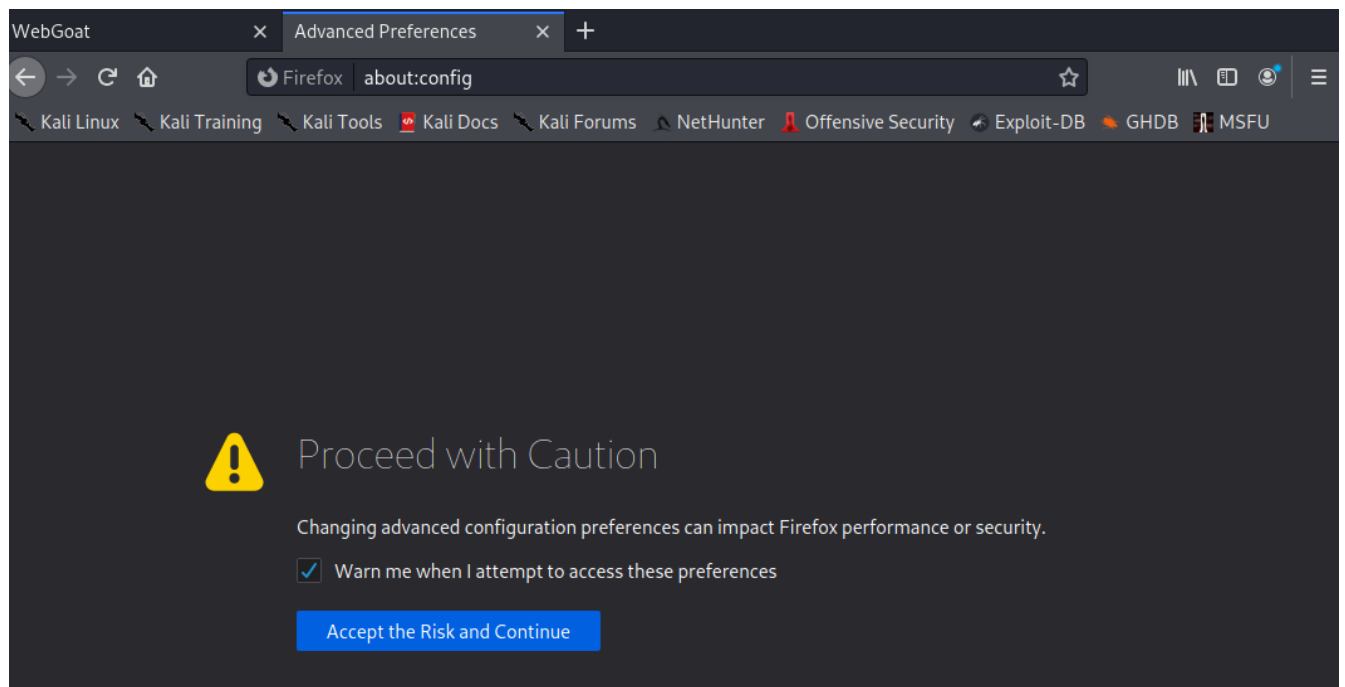
☐ Proxy DNS when using SOCKS v5

☐ Enable DNS over HTTPS

Use Provider Cloudflare (Default)

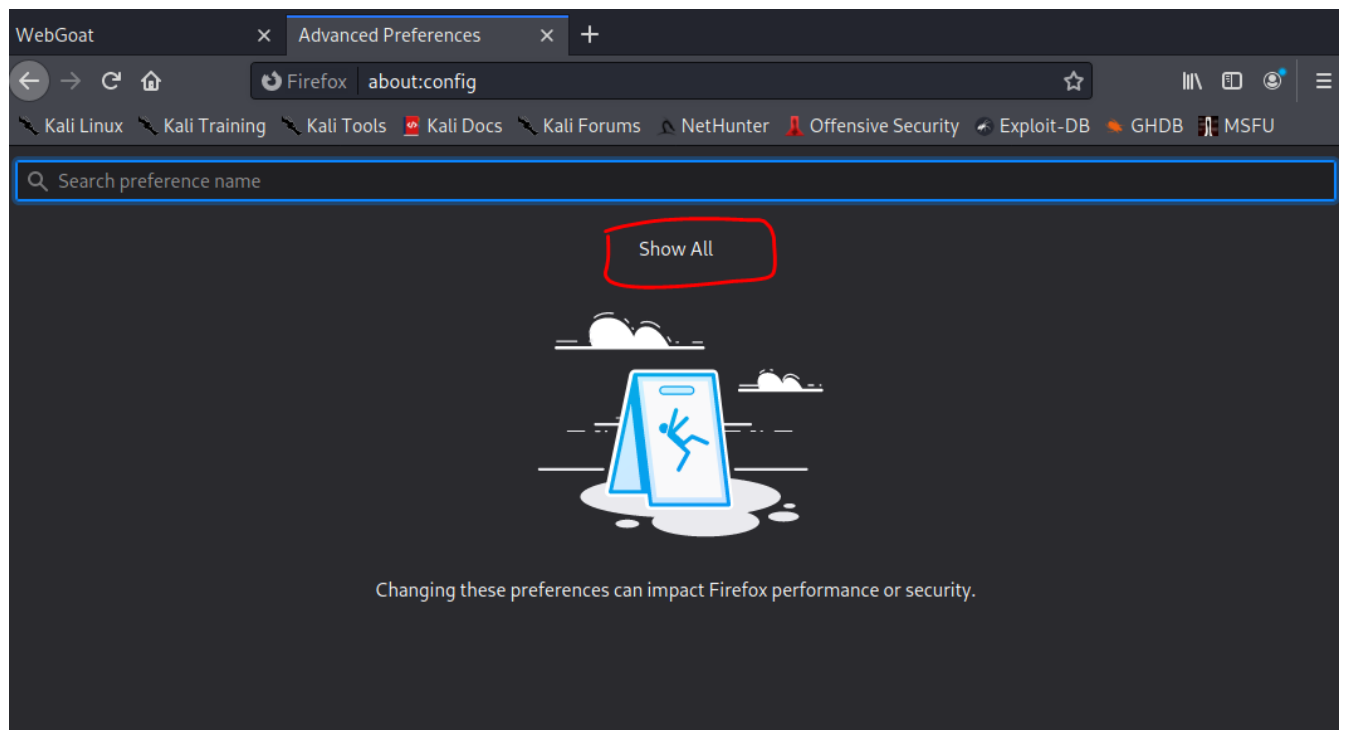
Help Cancel OK

In the browser URL location, type "**about:config**" and hit enter as shown below.

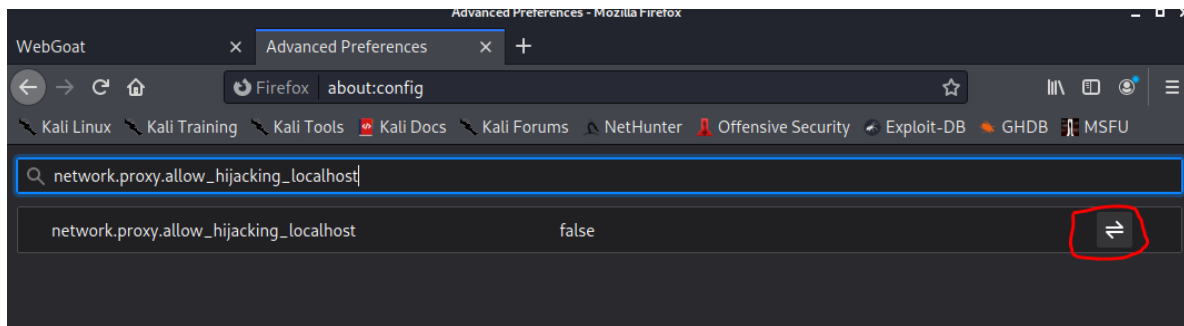


Click on "**Accept Risk and Continue**".

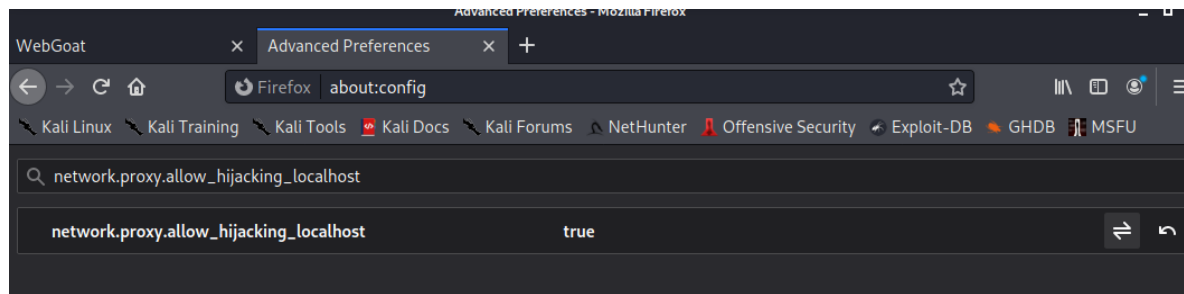
In the next window, click on "**Show All**".



In the search bar, type "**network.proxy.allow\_hijacking\_localhost**", as shown below. You will see the field set to "false". Click on the button (highlighted in red below) to switch it to "true".

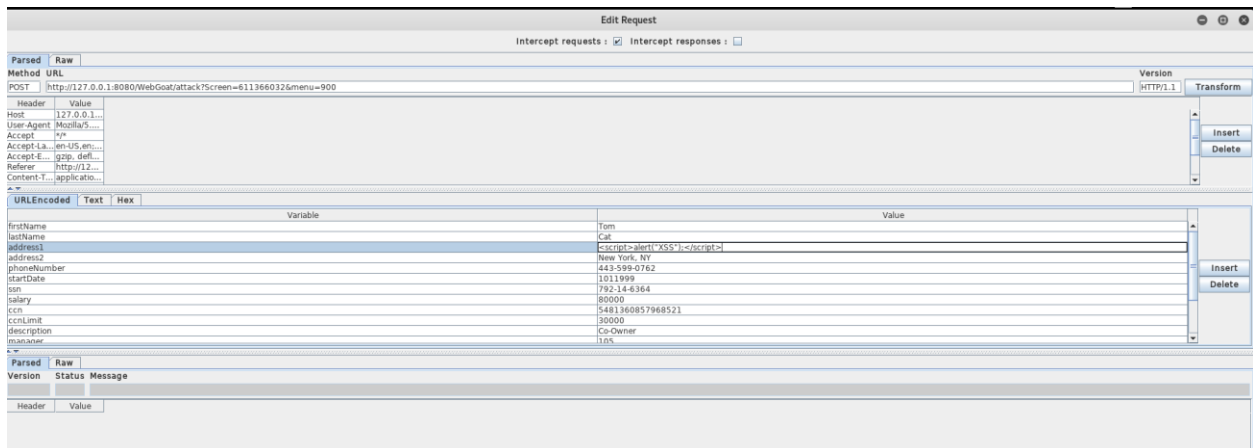


The field will now be set to "true".

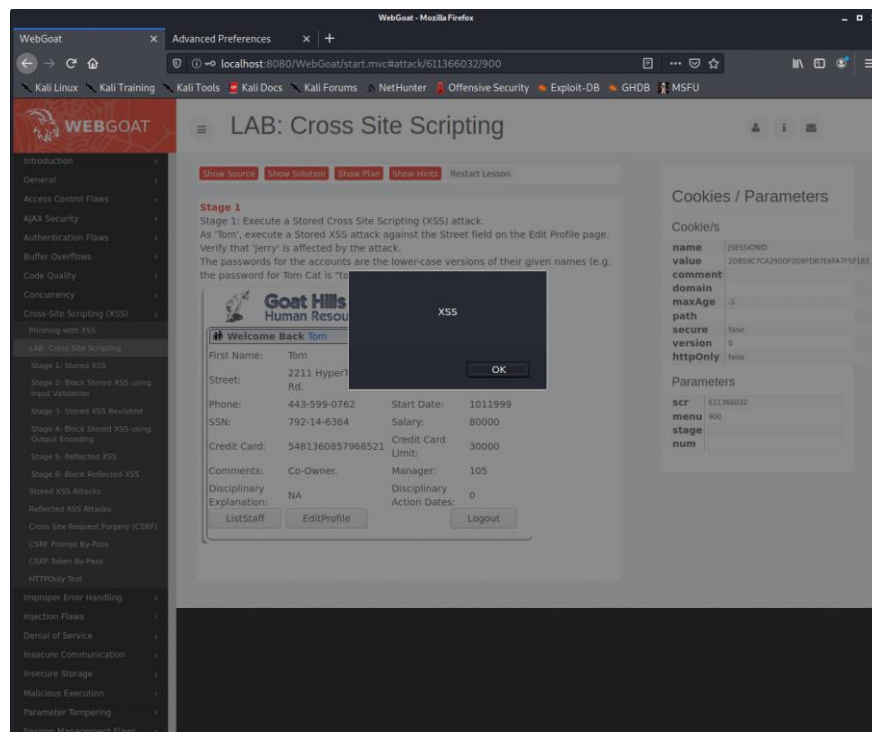


Now, click on **Update Profile** of Tom Cat (employee).

A window pops up showing all the fields in "Update Profile" form. Append the following JavaScript to the address1 field: **<script>alert("XSS");</script>**



Click **Accept Changes**.



Click **OK** on the XSS alert. Now, go to **webscarab** and uncheck the **Intercept requests** box under **Proxy** → **Manual Edit**.

Then, log out as Tom Cat (employee).

Now, let's login as another user and see if the JavaScript we have added works when we view Tom's profile.

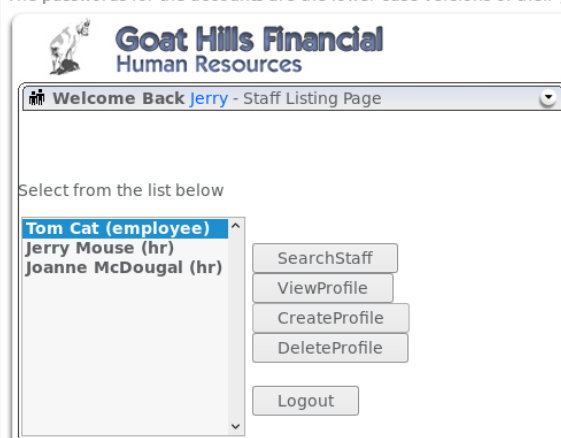
Login as Jerry Mouse (hr) where password is "jerry". Select Tom Cat (employee) and **View Profile**.

### 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").



We should see an alert dialog, saying "XSS". This confirms that our script runs when *another user* is logged in. Using this attack, an attacker can steal user information from the cookies of a victim.

Click **OK** on the XSS alert and logout as Jerry Mouse (hr).

Turn off intercept until needed to process the web requests without delay.

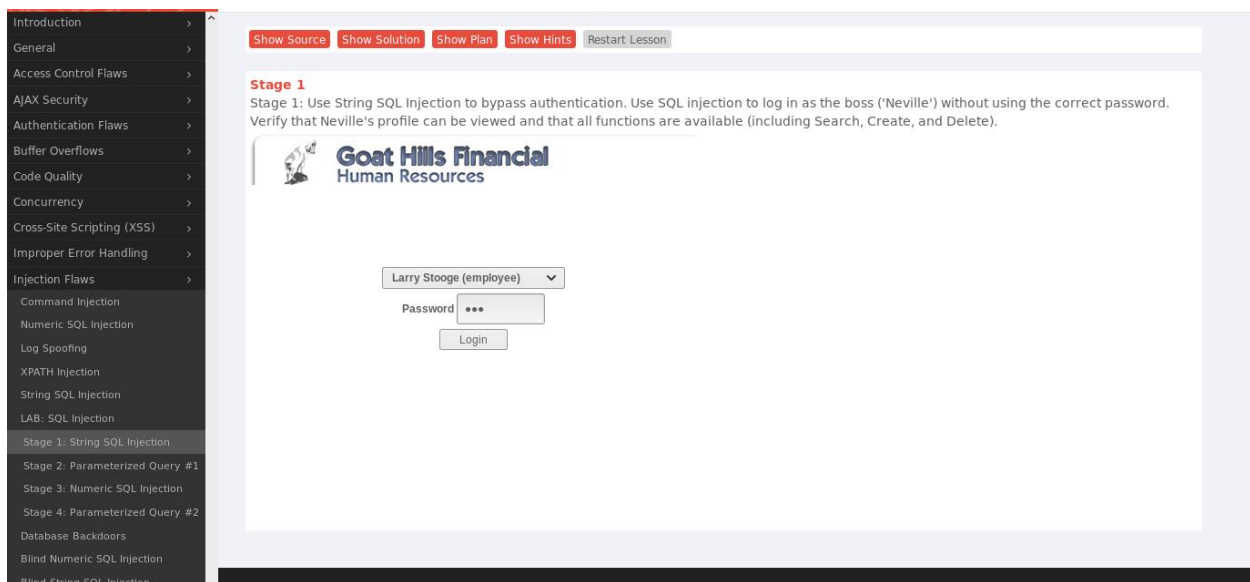
**Q6: Attach a screenshot of your results showing the lab was successful (with the XSS alert logged in as Jerry Mouse (hr)).**

## SQL INJECTION

SQL Injection is a technique to inject attack code into the SQL queries that run on the server. One simple attack is to inject an *attack string* to achieve unauthorized access to a user's account.

Select the following option in the WebGoat left hand menu:

**Injection Flaws → LAB: SQL Injection → Stage 1: String SQL Injection**



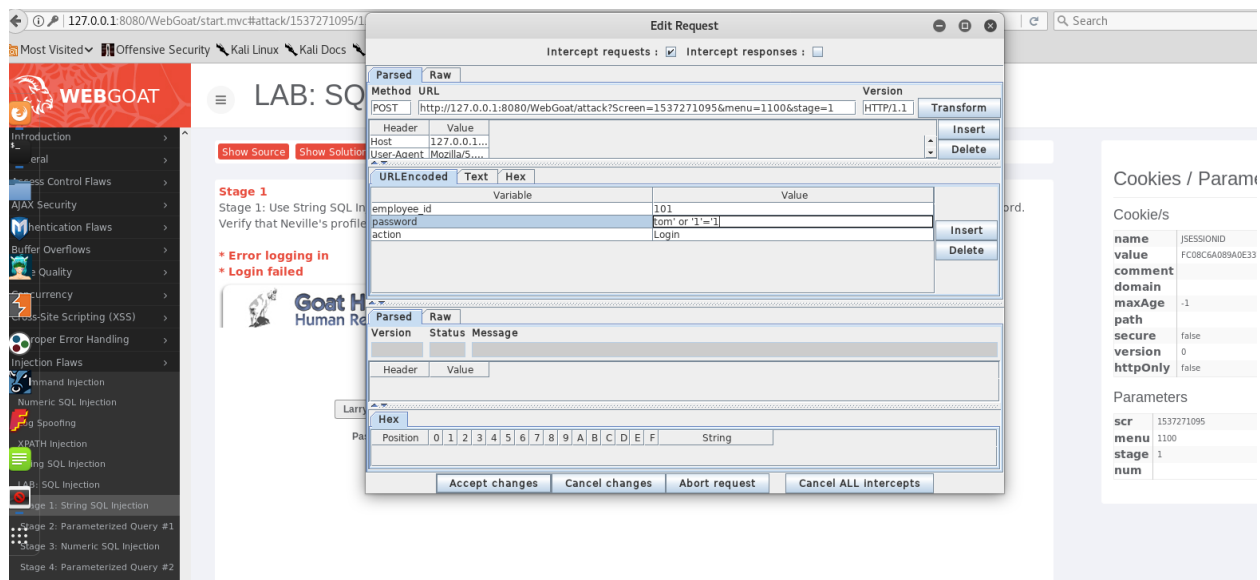
Start the intercept in webscarab:

Go to **Proxy → Manual Edit** and select **Intercept requests**.

In WebGoat, select the admin (last user on the list). We will see how an attacker can login to a vulnerable web application without providing the password. Click **Login** on the web application and this request is trapped the **webscarab** as follows:

Now tamper the password as follows:

Set password as **<any string>' OR '1'='1**. As an example, we set our password as: **PWNED' OR '1'='1**.



**Click Accept changes and Stop intercepting**

You should be now logged in as the administrator. To verify your admin rights, try viewing any profile.

Click **Logout** to logout as admin.

**Q7: Attach a screenshot of successful completion of task.**

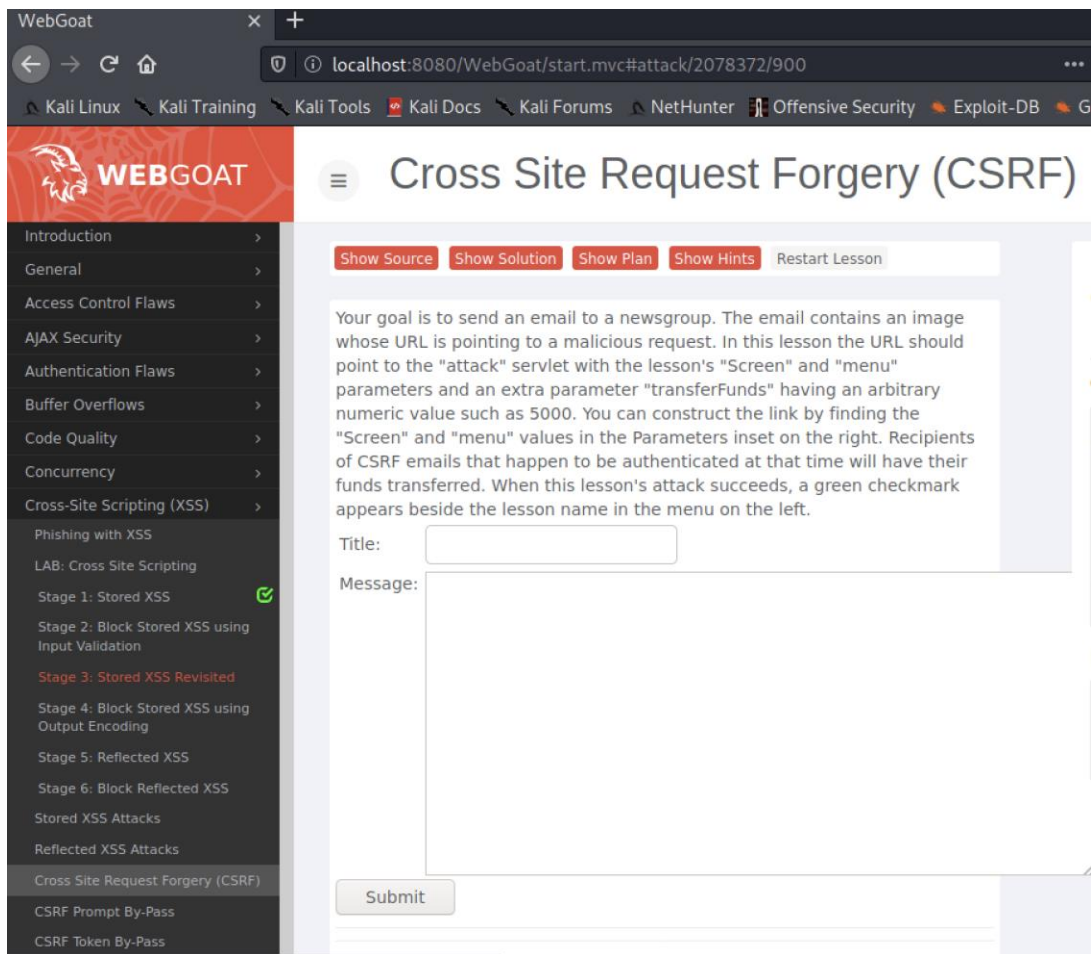
## CROSS SITE REQUEST FORGERY

CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated. With a little help of social engineering (like sending a link via email/chat), an attacker may force the users of a web application to execute actions of the attacker's choosing. A successful CSRF exploit can compromise end user data and operation in case of normal user. If the targeted end user is the administrator account, this can compromise the entire web application.

Select the following option in the WebGoat left hand menu:

**Cross Site Scripting (XSS) → Cross Site Request Forgery (CSRF)**





In the "**Title:**" field, enter a unique title to grab the victims' attention such as "**Free Kittens**".

Be sure the Title is unique by including your EUID so we know the submitted solution is yours.

In the "**Message:**" field, you will need to load an HTML image where the image is actually empty and a "transferFunds" parameter is loaded with an arbitrary value as follows:

```
<img src='attack?Screen=2078372&menu=900&transferFunds=5000'>
```

Then click the "Submit" button.

**Cross Site Request Forgery (CSRF)**

Show Source Show Solution Show Plan Show Hints Restart Lesson

Your goal is to send an email to a newsgroup. The email contains an image whose URL is pointing to a malicious request. In this lesson the URL should point to the "attack" servlet with the lesson's "Screen" and "menu" parameters and an extra parameter "transferFunds" having an arbitrary numeric value such as 5000. You can construct the link by finding the "Screen" and "menu" values in the Parameters inset on the right. Recipients of CSRF emails that happen to be authenticated at that time will have their funds transferred. When this lesson's attack succeeds, a green checkmark appears beside the lesson name in the menu on the left.

Title:

Message:

Submit

**Message List**

**Cookies / Parameters**

**Cookies**

name	value	comment	domain	maxAge	path	secure	version	httpOnly
JSESSIONID	780ASCEAC4A51F07005A365F90841F75			-1		false	0	false

**Parameters**

scr	2078372
menu	900
stage	
num	

(Note: the Screen and menu parameters come from the page source)

You should now scroll down to see your new entry in the Message List below:

pointing to a malicious request. In this lesson the URL should point to the "attack" servlet with the lesson's "Screen" and "menu" parameters and an extra parameter "transferFunds" having an arbitrary numeric value such as 5000. You can construct the link by finding the "Screen" and "menu" values in the Parameters inset on the right. Recipients of CSRF emails that happen to be authenticated at that time will have their funds transferred. When this lesson's attack succeeds, a green checkmark appears beside the lesson name in the menu on the left.

Title:

Message:

Submit

**Message List**

Free Kittens

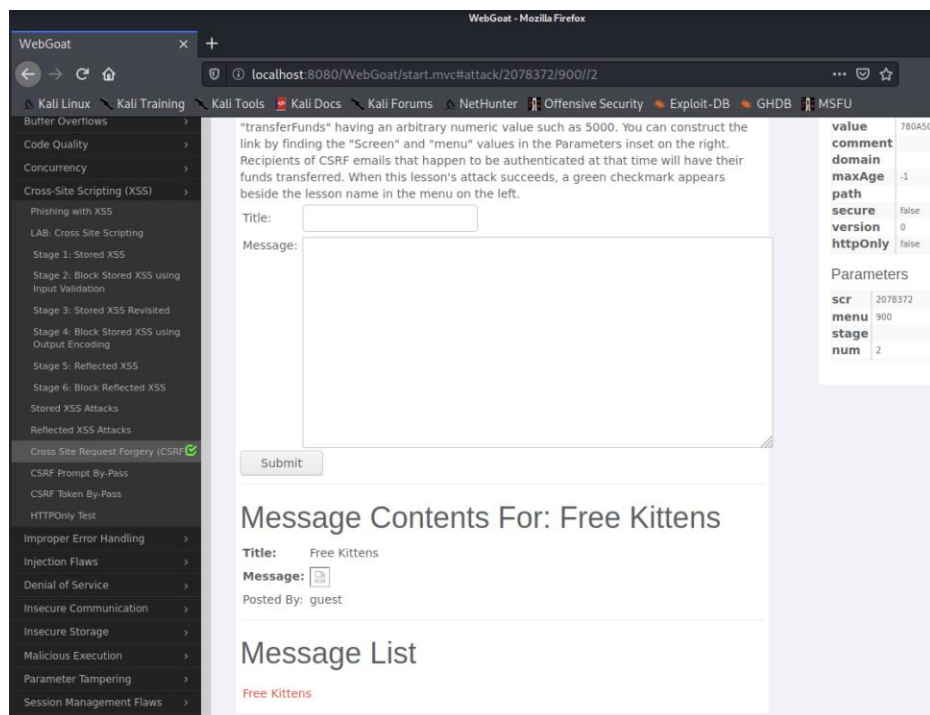
**Cookies**

name	value	comment	domain	maxAge	path	secure	version	httpOnly
JSESSIONID	780ASCEAC4A51F07005A365F90841F75			-1		false	0	false

**Parameters**

scr	2078372
menu	900
stage	
num	

Clicking on your message should cause your message to load on the victim webpage pulling in your exploit code and revealing a green check box next to the WebGoat Cross Site Request Forgery (CSRF) menu option revealing successful completion of this task. (You may need to refresh the page to get the check box.)



**Q8: Attach a screenshot showing the lab is successful.**

## BUFFER OVERFLOW

A buffer overflow occurs when a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold. Since buffers are created to contain a finite amount of data, the extra information - which has to go somewhere - can overflow into adjacent buffers, corrupting or overwriting the valid data held in them. Although it may occur accidentally through programming error, buffer overflow is an increasingly common type of security attack on data integrity. In buffer overflow attacks, the extra data may contain codes designed to trigger specific actions, in effect sending new instructions to the attacked computer that could, for example, damage the user's files, change data, or disclose confidential information. There are many variations of the buffer overflow but for this lab we will focus on web applications.

### Buffer Overflow and Web Applications

Attackers use buffer overflows to corrupt the execution stack of a web application. By sending carefully crafted input to a web application, an attacker can cause the web application to execute arbitrary code – effectively taking over the machine.

Buffer overflow flaws can be present in both the web server or application server products that serve the static and dynamic aspects of the site, or the web application itself. Buffer overflows found in widely used server products are likely to become widely known and can pose a significant

risk to users of these products. When web applications use libraries, such as a graphics library to generate images, they open themselves to potential buffer overflow attacks.

Buffer overflows can also be found in custom web application code, and may even be more likely given the lack of scrutiny that web applications typically go through. Buffer overflow flaws in custom web applications are less likely to be detected because there will normally be far fewer hackers trying to find and exploit such flaws in a specific application. If discovered in a custom application, the ability to exploit the flaw (other than to crash the application) is significantly reduced by the fact that the source code and detailed error messages for the application are normally not available to the hacker.

## Consequences

- **Category: Availability:** Buffer overflows generally lead to crashes. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop.
- **Access control (instruction processing):** Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy.
- **Other:** When the consequence is arbitrary code execution, this can often be used to subvert any other security service.

## Exposure period

- **Requirements specification:** The choice could be made to use a language that is not susceptible to these issues.
- **Design:** Mitigating technologies such as safe-string libraries and container abstractions could be introduced.
- **Implementation:** Many logic errors can lead to this condition. It can be exacerbated by lack of or misuse of mitigating technologies.

Select the following option in the WebGoat left hand menu:

## Buffer Overflows → Off-by-One Overflows

A web page will be displayed; the web page is simulating a room booking.

The screenshot shows the WebGoat interface for the 'Off-by-One Overflows' challenge. The main content area displays a login form for the 'OWASP Hotel'. The form includes three input fields: 'First Name', 'Last Name', and 'Room Number', each followed by an asterisk indicating it is required. A 'Submit' button is located below the 'Room Number' field. A message at the bottom of the form states: '\* The above fields are required for login.' The left sidebar contains a list of challenges, with 'Off-by-One Overflows' selected. The right sidebar shows 'Cookies / Parameters' with a table of cookies and a table of parameters.

name	value
SESSIONID	FC08CA09ADE337DE7364808CEB08E
comment	
domain	-.1
maxAge	
path	
secure	false
version	0
httpOnly	false

name	value
scr	136032128
menu	600
stage	
num	

What you need to do

1. You will enter arbitrary details.

First Name	A
Last Name	A
Room Number	Over 6000 A's or any characters as shown below

WEBGOAT Off-by-One Overflows

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:  +

Last Name:  +

Room Number:  +

\* The above fields are required for login.

Cookies / Parameters

Cookie/s

name	value	comment
SESSIONID	E9KCBFF3A51974F6E27DD5FE3B83693	
domain		
maxAge	-1	
path		
secure	false	
version	0	
httpOnly	false	

Parameters

scr	736032126
menu	600
stage	
num	

2. Click "**Submit**" and then "**Accept Terms**".

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

Session Management Flaws

Web Services

Admin Functions

Challenge

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?

\* 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.

We would like to thank you for your payment.

Cookies / Parameters

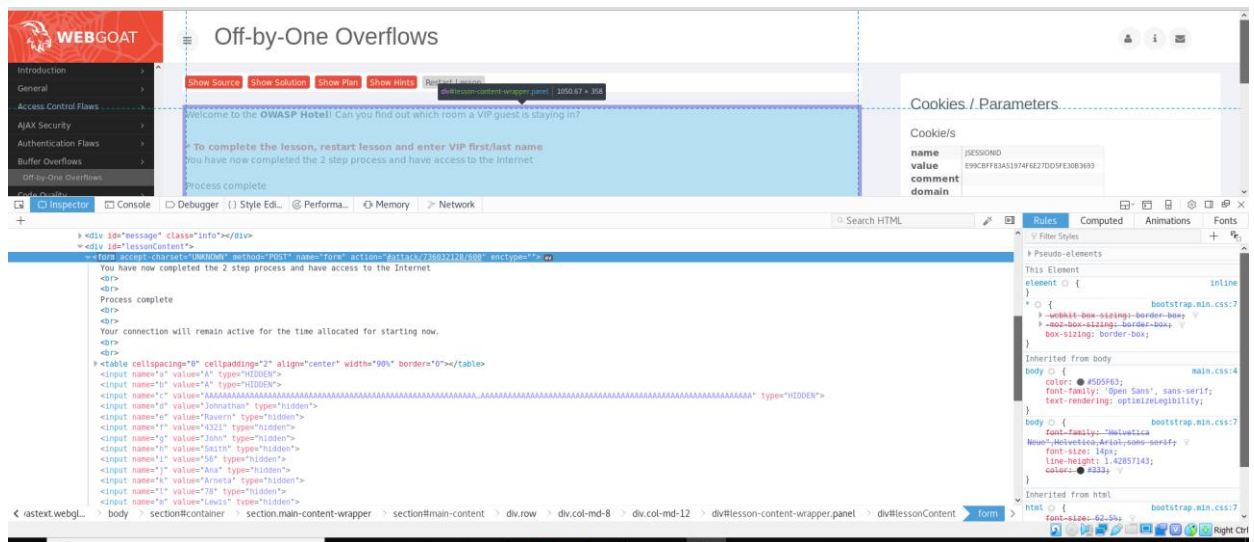
Cookie/s

name	value	comment
SESSIONID	E9KCBFF3A51974F6E27DD5FE3B83693	
domain		
maxAge	-1	
path		
secure	false	
version	0	
httpOnly	false	

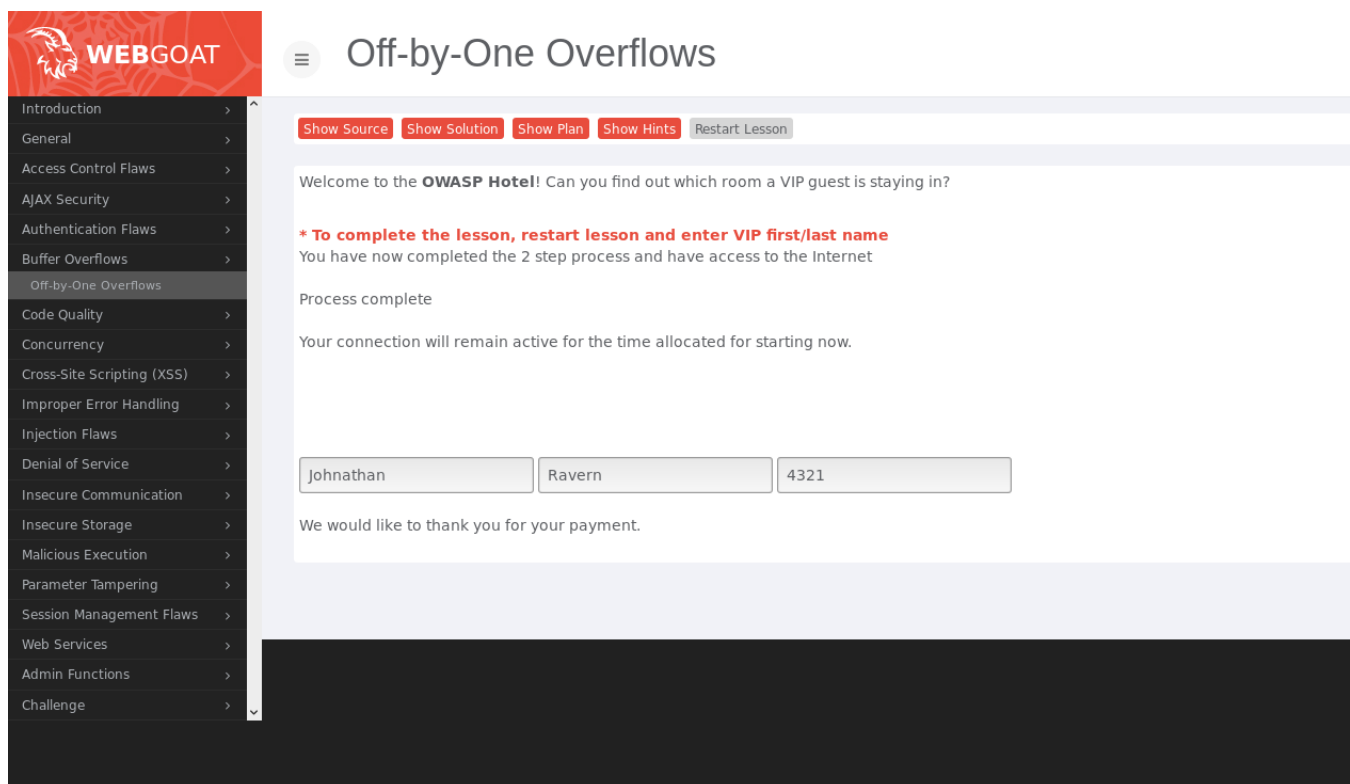
Parameters

scr	736032126
menu	600
stage	
num	

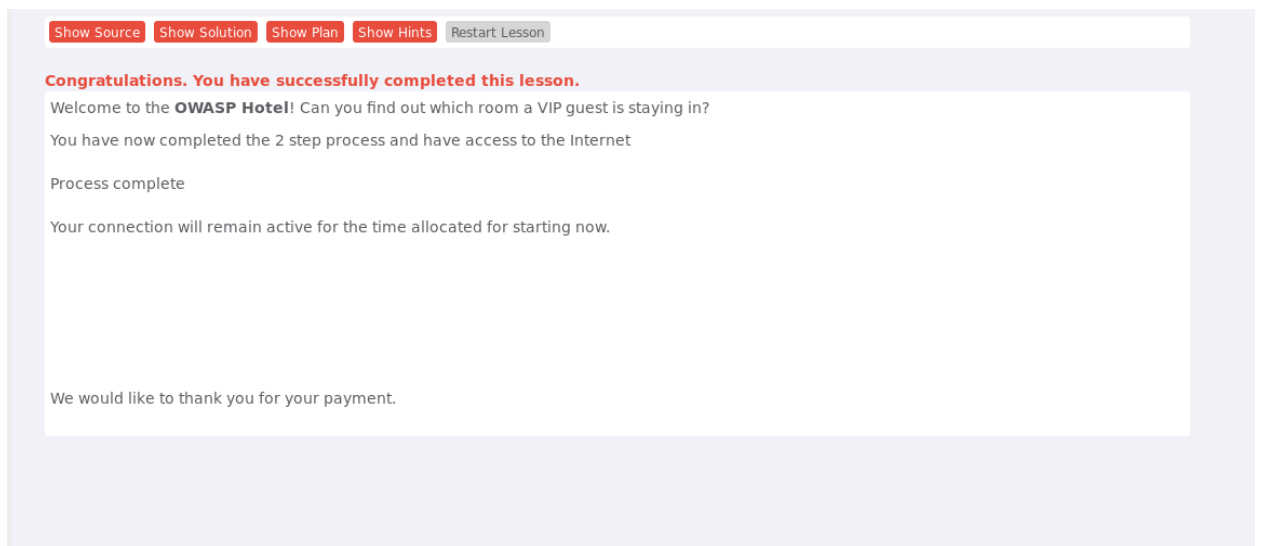
3. Right click on the page and select "**Inspect Element (Q)**" you will be able to see hidden form fields. Remove the `type="HIDDEN"` parameters for all of the guests.



4. After deleting the parameter `type="HIDDEN"` you will be able to see details of your input data on the web page. Remember the other guest's information (i.e., First Name, Last Name, Room Number).



5. Click on "**Restart Lesson**" and give these room details for one of the users to complete the task. (You can use any of the other member room details.)



**Q9: What is John Smith's room number? Attach a screenshot of the web page showing the details.**