Karan Tank

COMP 357

LAB 4

WEB AUTHENTICATION AND AUTHORIZATION

Contents

Access Control Flaws

CVE-2019-9730

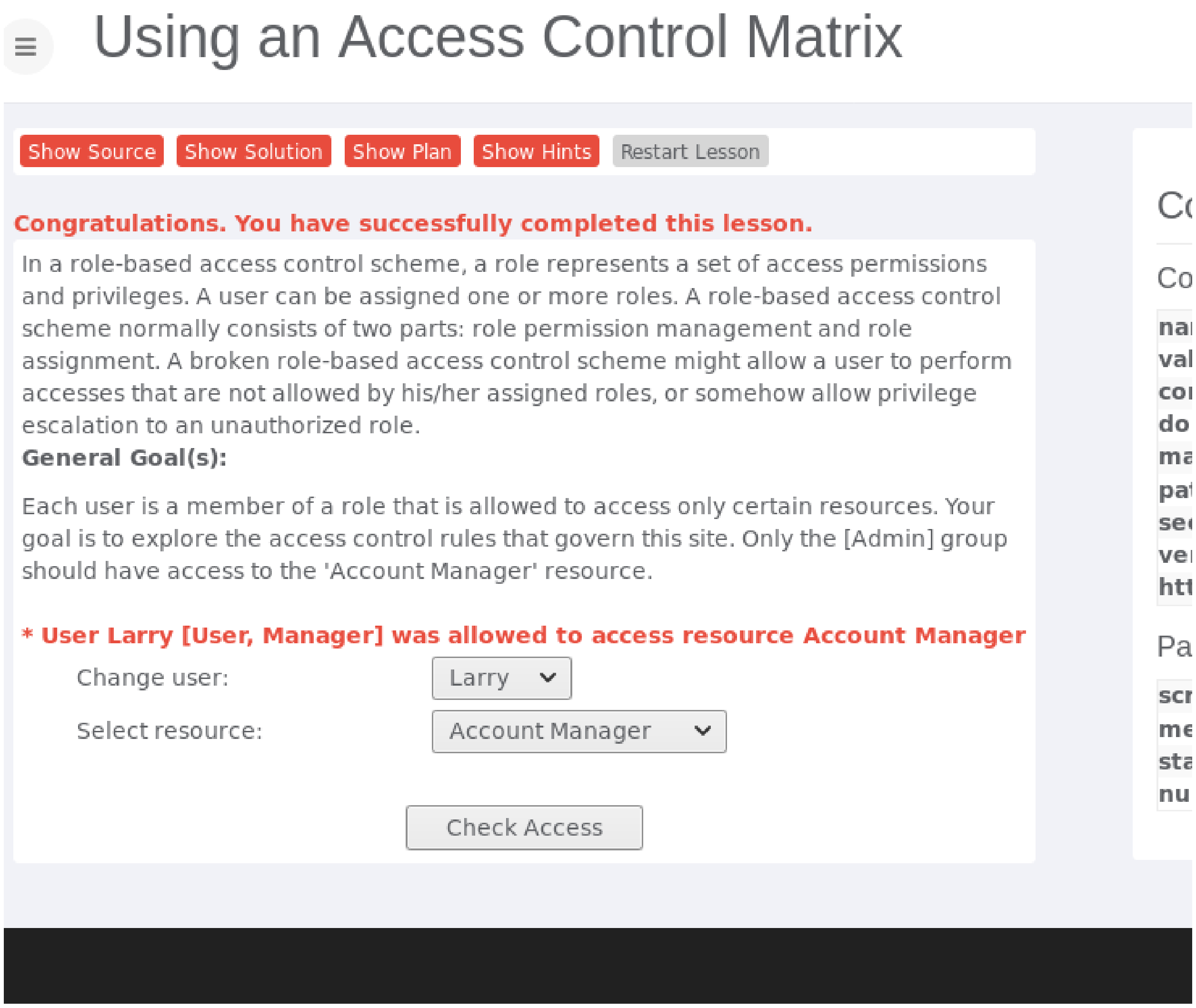
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9730

This Vulnerability is incorrect access control in the CxUtilsvc.exe component of the synaptic audio driver could allow a standard user to increase access privileges to the windows registry via an unpublished api.This was a high severity vulnerability.

It can be mitigated by updating synaptics to newer or indicated for your model in the product Impact section.

Access Control Matrix

- In this part, we will have to explore the broken web application so that we allow user with admin privileges which are represented by “Account Manager”
- A simple approach is to select users in the ‘change user’ field and keep the ‘Account manager’ constant.
- After multiple tries, it will complete the lesson with the combination of Larry as account manager



Using an Access Control Matrix

Congratulations. You have successfully completed this lesson.

In a role-based access control scheme, a role represents a set of access permissions and privileges. A user can be assigned one or more roles. A role-based access control scheme normally consists of two parts: role permission management and role assignment. A broken role-based access control scheme might allow a user to perform accesses that are not allowed by his/her assigned roles, or somehow allow privilege escalation to an unauthorized role.

General Goal(s):

Each user is a member of a role that is allowed to access only certain resources. Your goal is to explore the access control rules that govern this site. Only the [Admin] group should have access to the 'Account Manager' resource.

*** User Larry [User, Manager] was allowed to access resource Account Manager**

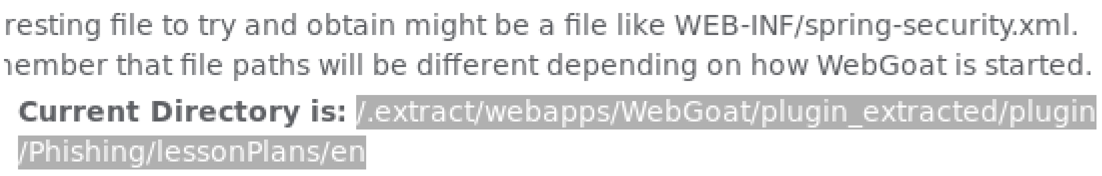
Change user: Larry

Select resource: Account Manager

Figure 1

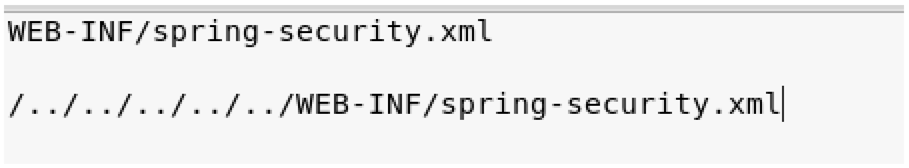
Bypass a path-based access control scheme

- The goal of this lab is to access a webpage which is not allowed for the ‘guest’ user to view
- We are given a list of files to select from and can be viewed by guest user
- In this the vulnerability is that the file parameter allows to include special characters using which the other directory file path can be accessed
- We will first open any file
- After that we will replace the whole file path with “../”
- And then we will add the path of desired file


resting file to try and obtain might be a file like WEB-INF/spring-security.xml.
ıember that file paths will be different depending on how WebGoat is started.

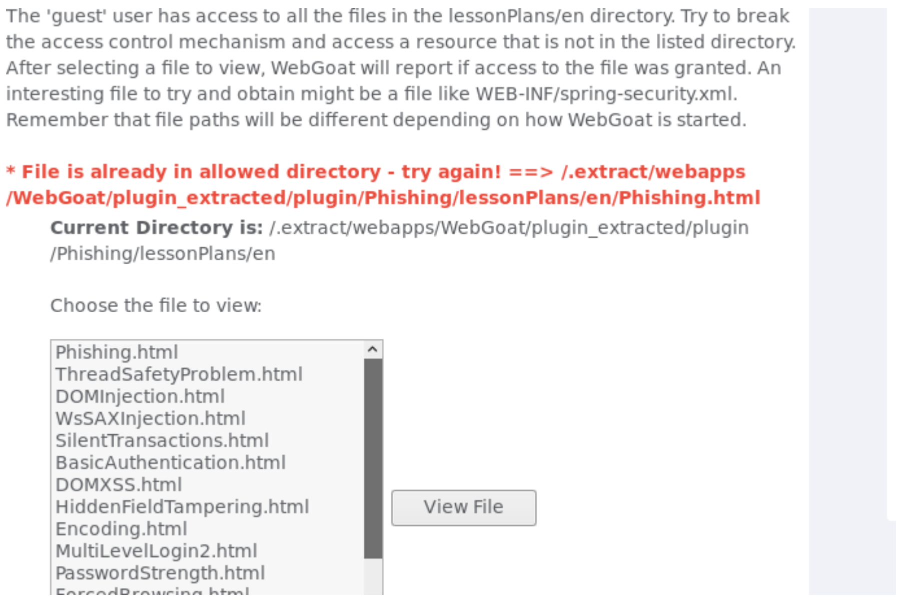
Current Directory is: /.extract/webapps/WebGoat/plugin_extracted/plugin/Phishing/lessonPlans/en

Figure 2



```
WEB-INF/spring-security.xml

/../../../../../WEB-INF/spring-security.xml
```

Figure 3

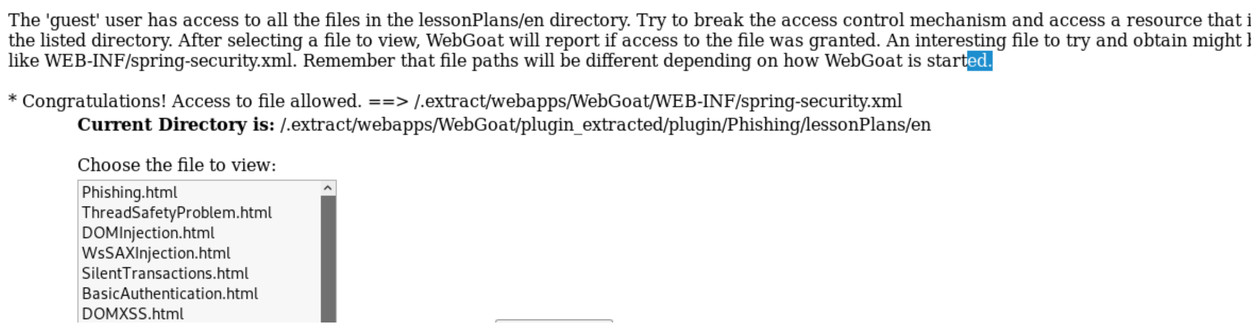

The 'guest' user has access to all the files in the lessonPlans/en directory. Try to break the access control mechanism and access a resource that is not in the listed directory. After selecting a file to view, WebGoat will report if access to the file was granted. An interesting file to try and obtain might be a file like WEB-INF/spring-security.xml. Remember that file paths will be different depending on how WebGoat is started.

*** File is already in allowed directory - try again! ==> /.extract/webapps/WebGoat/plugin_extracted/plugin/Phishing/lessonPlans/en/Phishing.html**

Current Directory is: /.extract/webapps/WebGoat/plugin_extracted/plugin/Phishing/lessonPlans/en

Choose the file to view:

Figure 4


The 'guest' user has access to all the files in the lessonPlans/en directory. Try to break the access control mechanism and access a resource that i
the listed directory. After selecting a file to view, WebGoat will report if access to the file was granted. An interesting file to try and obtain might l
like WEB-INF/spring-security.xml. Remember that file paths will be different depending on how WebGoat is started.

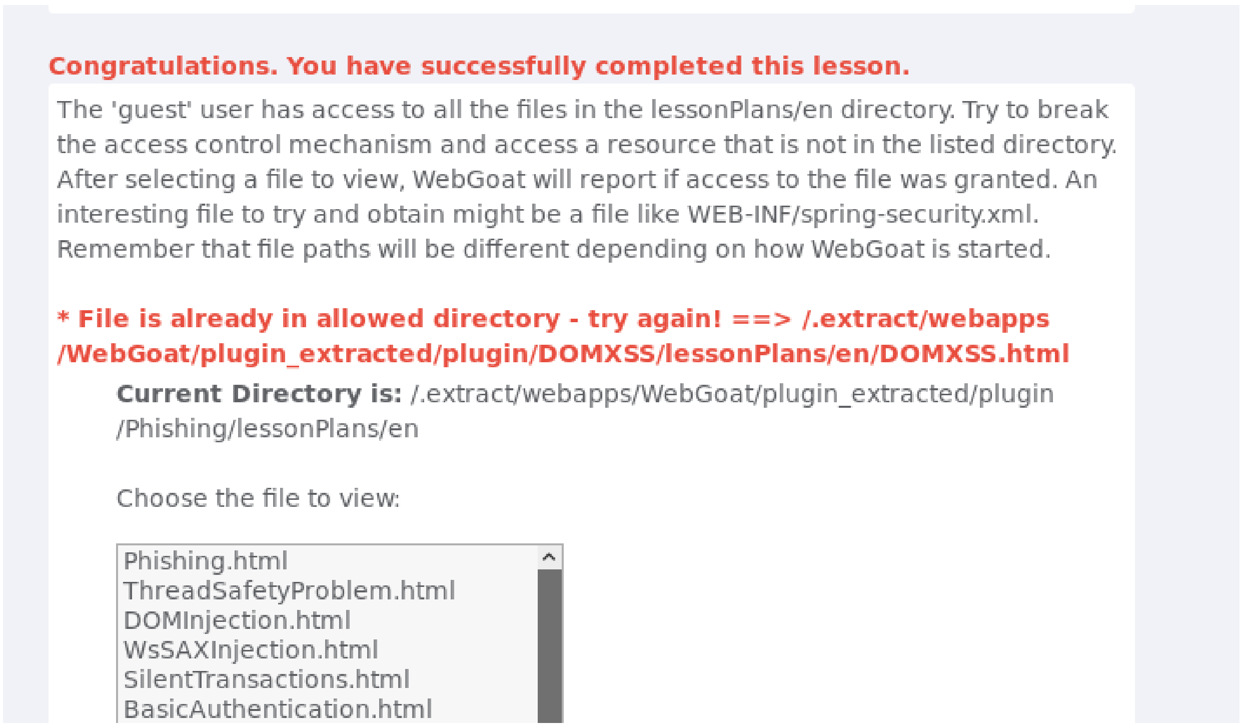
* Congratulations! Access to file allowed. ==> /.extract/webapps/WebGoat/WEB-INF/spring-security.xml

Current Directory is: /.extract/webapps/WebGoat/plugin_extracted/plugin/Phishing/lessonPlans/en

Choose the file to view:

```
Phishing.html
ThreadSafetyProblem.html
DOMInjection.html
WsSAXInjection.html
SilentTransactions.html
BasicAuthentication.html
DOMXSS.html
```

Figure 5


Congratulations. You have successfully completed this lesson.

The 'guest' user has access to all the files in the lessonPlans/en directory. Try to break the access control mechanism and access a resource that is not in the listed directory. After selecting a file to view, WebGoat will report if access to the file was granted. An interesting file to try and obtain might be a file like WEB-INF/spring-security.xml. Remember that file paths will be different depending on how WebGoat is started.

* File is already in allowed directory - try again! ==> /.extract/webapps/WebGoat/plugin_extracted/plugin/DOMXSS/lessonPlans/en/DOMXSS.html

Current Directory is: /.extract/webapps/WebGoat/plugin_extracted/plugin/Phishing/lessonPlans/en

Choose the file to view:

```
Phishing.html
ThreadSafetyProblem.html
DOMInjection.html
WsSAXInjection.html
SilentTransactions.html
BasicAuthentication.html
```

Figure 6

CVE-2017-9502

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9502

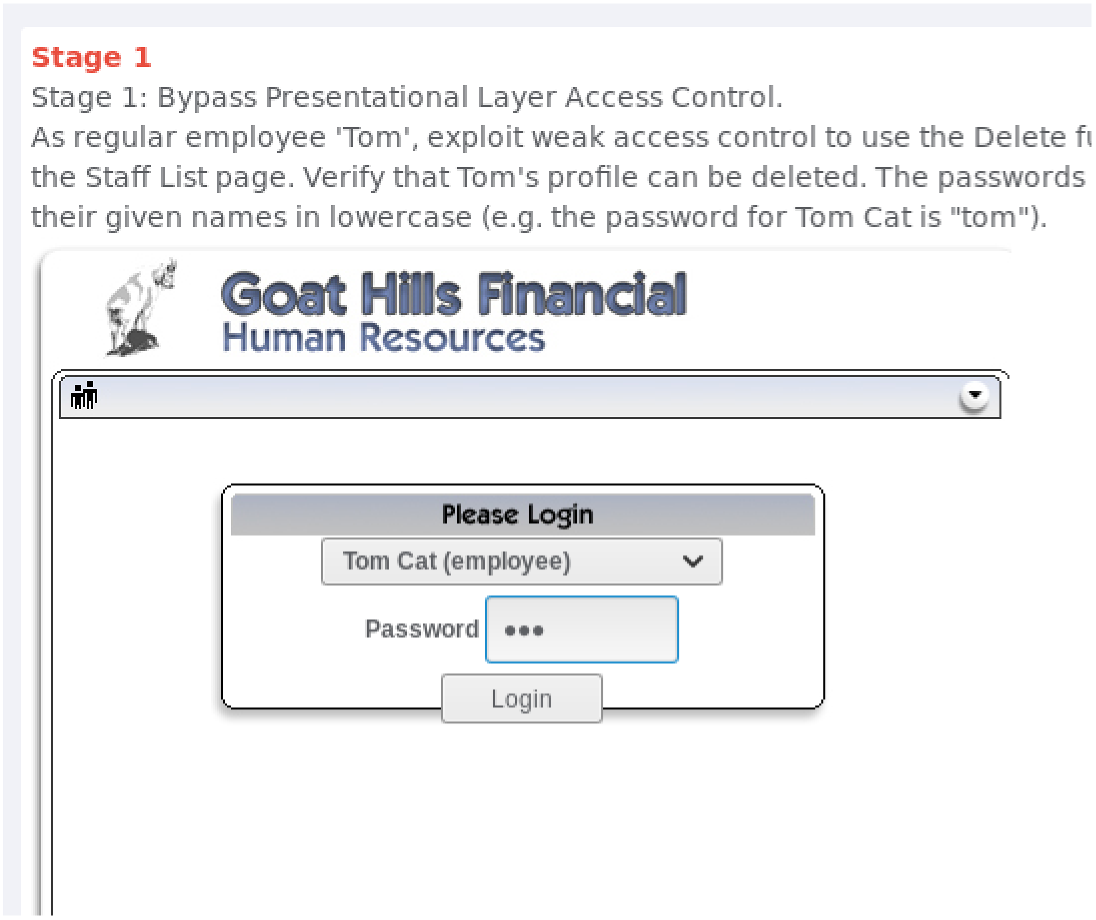
The vulnerability was found in curl before 7.54.1 on windows and DOS. When libcurl is given a file, the url that doesn't use two slashes following colon or is told that the file is the default scheme to use URL's without scheme and if the given path starts with a drive letter, then the libcurl would copy the path with wrong offset so that end of the given path would write beyond the malloc buffer up to seven bytes.

There were no exploits of this flaw.

Patch is available on this link

Stage 1

- In this part of the lab, we are to delete the user named Tom. It has weak exploit control which will be used to delete from the staff list page.
- For this we will first need to login to TOM.
- Username : tom password tom is given
- After that, we will start the intercept on the burp suite and click on view profile
- We will then change the action from viewprofile to Deleteprofile and click on → forward
- It will forward the request and will delete the user named tom from the staff list



Stage 1

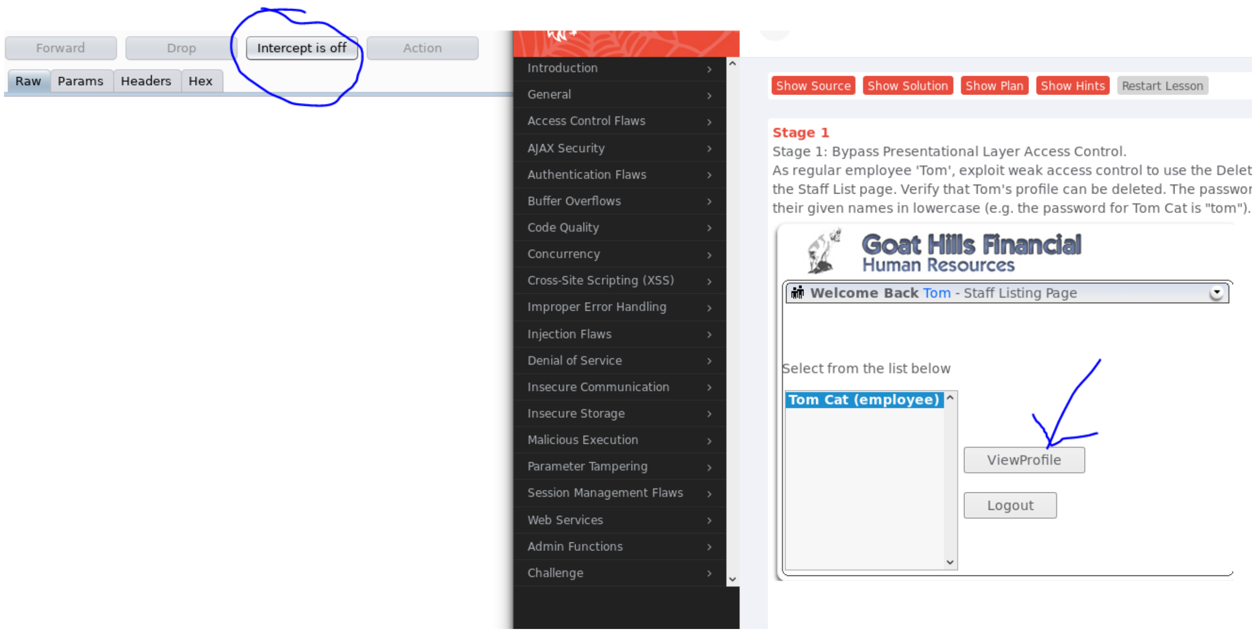
Stage 1: Bypass Presentational Layer Access Control.

As regular employee 'Tom', exploit weak access control to use the Delete fu

the Staff List page. Verify that Tom's profile can be deleted. The passwords

their given names in lowercase (e.g. the password for Tom Cat is "tom").

Figure 7



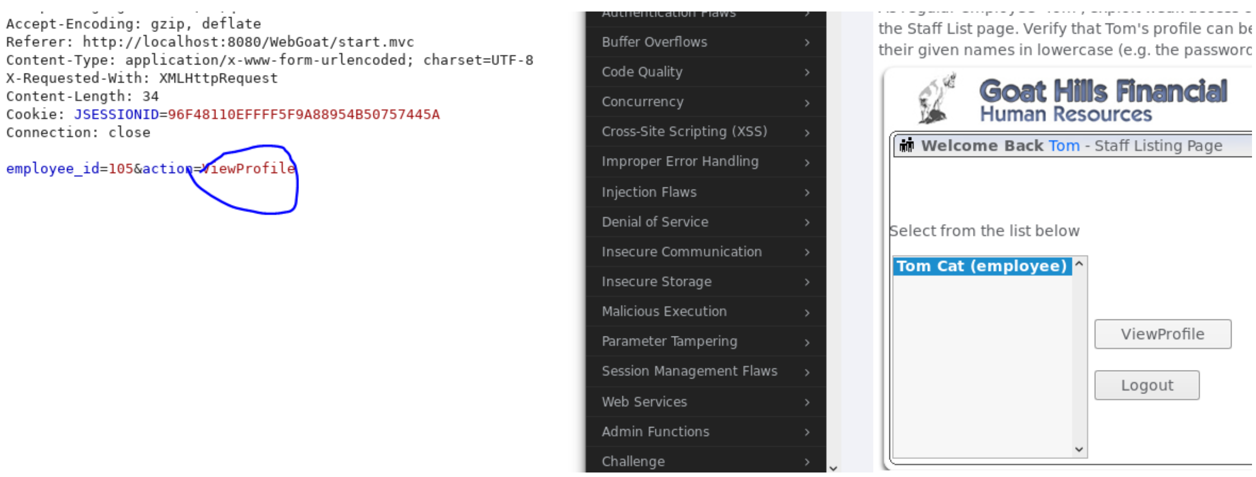
Stage 1
Stage 1: Bypass Presentational Layer Access Control.
As regular employee 'Tom', exploit weak access control to use the Delet
the Staff List page. Verify that Tom's profile can be deleted. The passwor
their given names in lowercase (e.g. the password for Tom Cat is "tom").

Welcome Back Tom - Staff Listing Page

Select from the list below

Tom Cat (employee)

Figure 8


```
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 34
Cookie: JSESSIONID=96F48110EFFFF5F9A88954B50757445A
Connection: close

employee_id=105&action=ViewProfile
```

the Staff List page. Verify that Tom's profile can be
their given names in lowercase (e.g. the password

Welcome Back Tom - Staff Listing Page

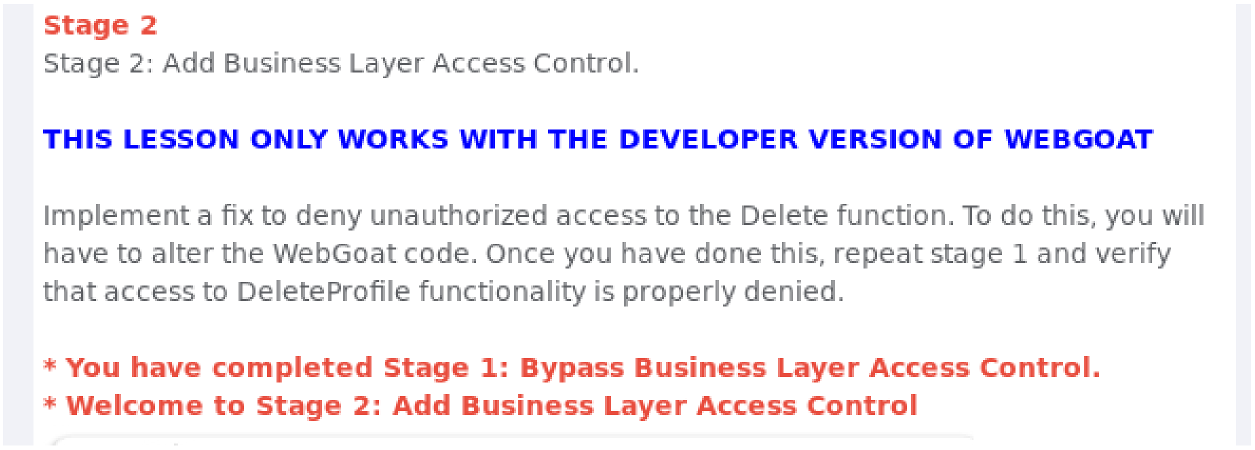
Select from the list below

Tom Cat (employee)

Figure 9

```
employee_id=105&action=DeleteProfile
```

Figure 10



Stage 2

Stage 2: Add Business Layer Access Control.

THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

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

*** You have completed Stage 1: Bypass Business Layer Access Control.**
*** Welcome to Stage 2: Add Business Layer Access Control**

Figure 11

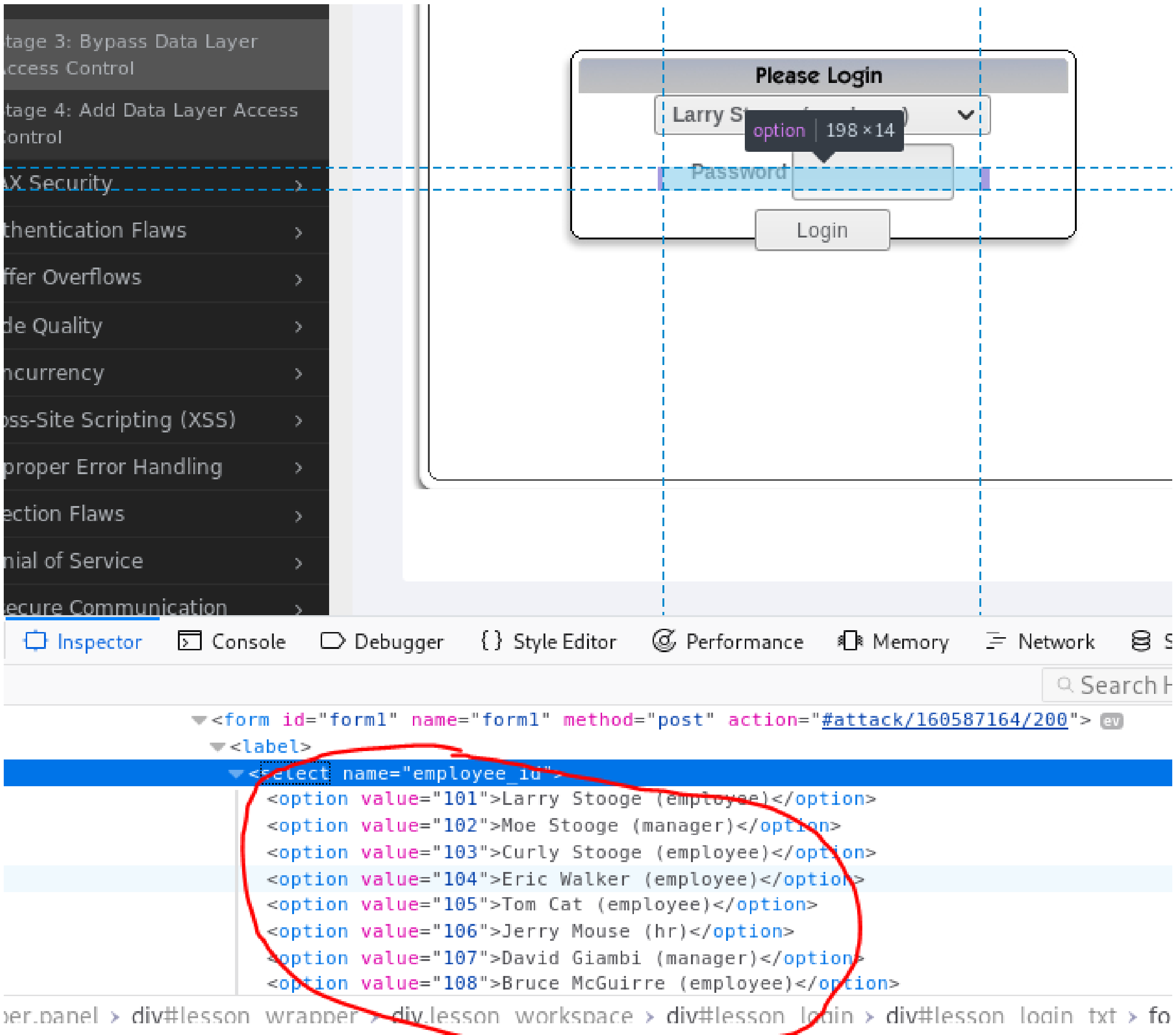
CVE-2019-1590

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-1590

This Vulnerability was found in transport layer security (TLS) protocol certificate validation functionality of cisco nexus 9000 series application centric infrastructure mode switch software. It could a remote hacker to perform insecure tls client authentication on an affected device. An attacker who has possession of a certificate can exploit this vulnerability by presenting a valid certificate while attempting to connect the target device.

Stage 3

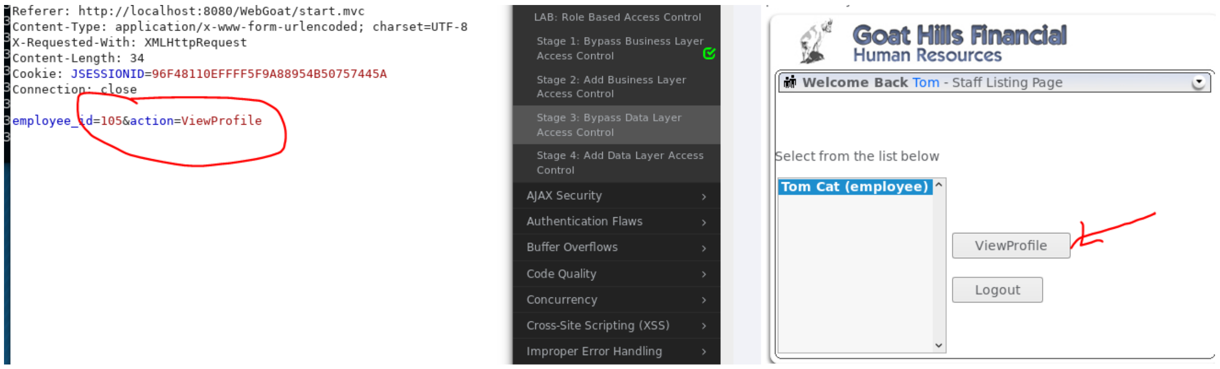
- For this part of lab, we will have to view to some other user using the user id and password we have
- For this, first we will open the inspect element and click on the user, you will get the user names and user id of everyone
- Choose a user id you want
- Now, login to the user you have which is tom and password tom
- Now start the intercept of the burp suite and click on view profile
- Now, in the intercept, change the id to your desired id and you can see that persons profile.



```
<form id="form1" name="form1" method="post" action="#attack/160587164/200"> ev
  <label>
    <select name="employee_id">
      <option value="101">Larry Stooge (employee)</option>
      <option value="102">Moe Stooge (manager)</option>
      <option value="103">Curly Stooge (employee)</option>
      <option value="104">Eric Walker (employee)</option>
      <option value="105">Tom Cat (employee)</option>
      <option value="106">Jerry Mouse (hr)</option>
      <option value="107">David Giambi (manager)</option>
      <option value="108">Bruce McGuirre (employee)</option>
```

er.panel › div#lesson_wrapper › div.lesson_workspace › div#lesson_login › div#lesson_login_txt › fo

Figure 12



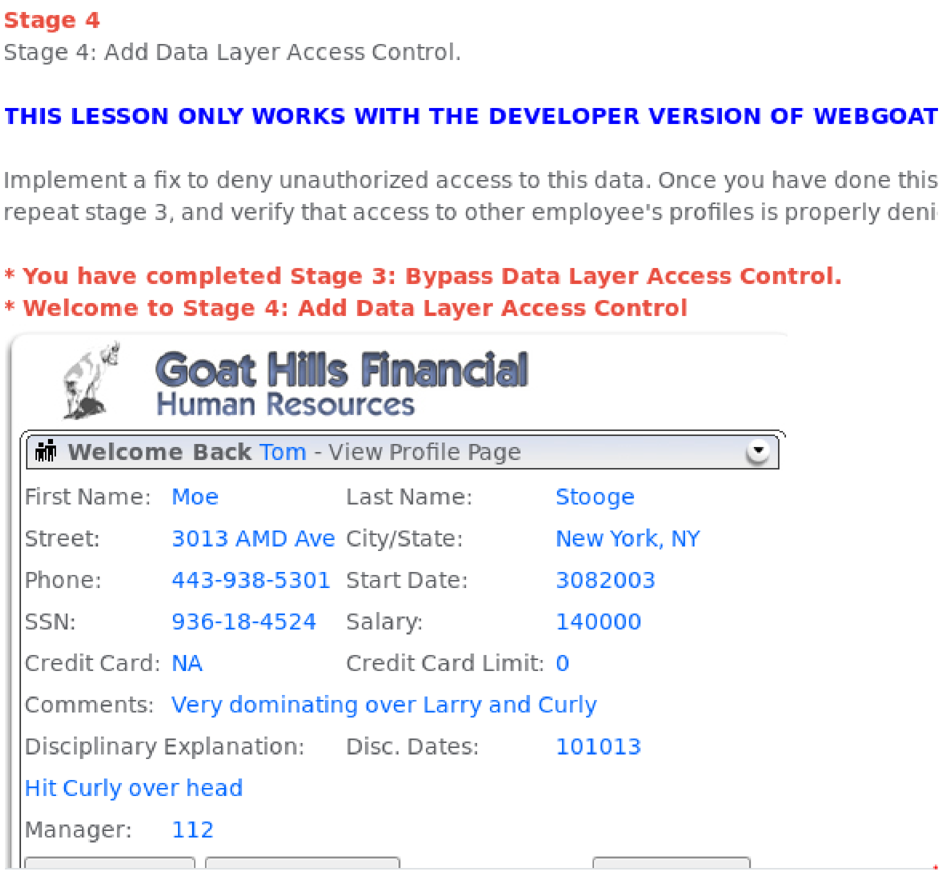
```
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 34
Cookie: JSESSIONID=96F48110EFFFF5F9A88954B50757445A
Connection: close

employee_id=105&action=ViewProfile
```

LAB: Role Based Access Control
Stage 1: Bypass Business Layer Access Control
Stage 2: Add Business Layer Access Control
Stage 3: Bypass Data Layer Access Control
Stage 4: Add Data Layer Access Control
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Goat Hills Financial
Human Resources
Welcome Back Tom - Staff Listing Page
Select from the list below
Tom Cat (employee)
ViewProfile
Logout

Figure 13

```
employee_id=102&action=ViewProfile
```

Figure 14



Stage 4

Stage 4: Add Data Layer Access Control.

THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT

Implement a fix to deny unauthorized access to this data. Once you have done this repeat stage 3, and verify that access to other employee's profiles is properly deni

*** You have completed Stage 3: Bypass Data Layer Access Control.**
*** Welcome to Stage 4: Add Data Layer Access Control**

First Name:	Moe	Last Name:	Stooge
Street:	3013 AMD Ave	City/State:	New York, NY
Phone:	443-938-5301	Start Date:	3082003
SSN:	936-18-4524	Salary:	140000
Credit Card:	NA	Credit Card Limit:	0
Comments:	Very dominating over Larry and Curly		
Disciplinary Explanation:		Disc. Dates:	101013
Hit Curly over head			
Manager:	112		

Figure 15

Authentication Flaws

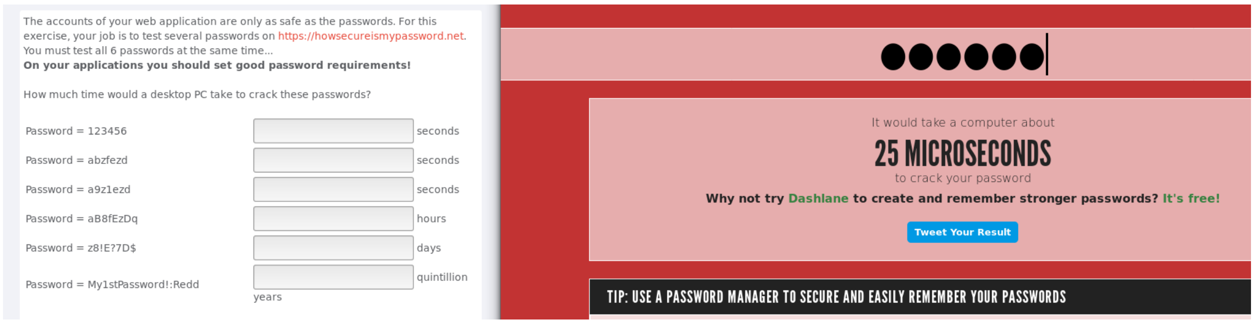
CVE-2017-10000247

https://codeigniter.com/userguide3/changelog.html#version-3-1-4

This vulnerability was found in the British Colombia Institute of Technology CodeIgniter 3.1.3 is Vulnerable to http header Injection in the set_status_header() common function under apache resulting in HTTP Header Injection flaws. You can mitigate this vulnerability by updating to newer versions which fixes the header injection vulnerability, fixed byte-safety issues in encryption library when mbstring.func_overload is enabled.

Password Strength

- For this part of the lab, we will enter the passwords given to the website and it will give us the time it will crack the password.



The accounts of your web application are only as safe as the passwords. For this exercise, your job is to test several passwords on https://howsecureismypassword.net. You must test all 6 passwords at the same time...
On your applications you should set good password requirements!

How much time would a desktop PC take to crack these passwords?

Password = 123456		seconds
Password = abzfezd		seconds
Password = a9z1ezd		seconds
Password = aB8fEzDq		hours
Password = z8!E?7D$		days
Password = My1stPassword!:Redd		quintillion years

●●●●●●

It would take a computer about

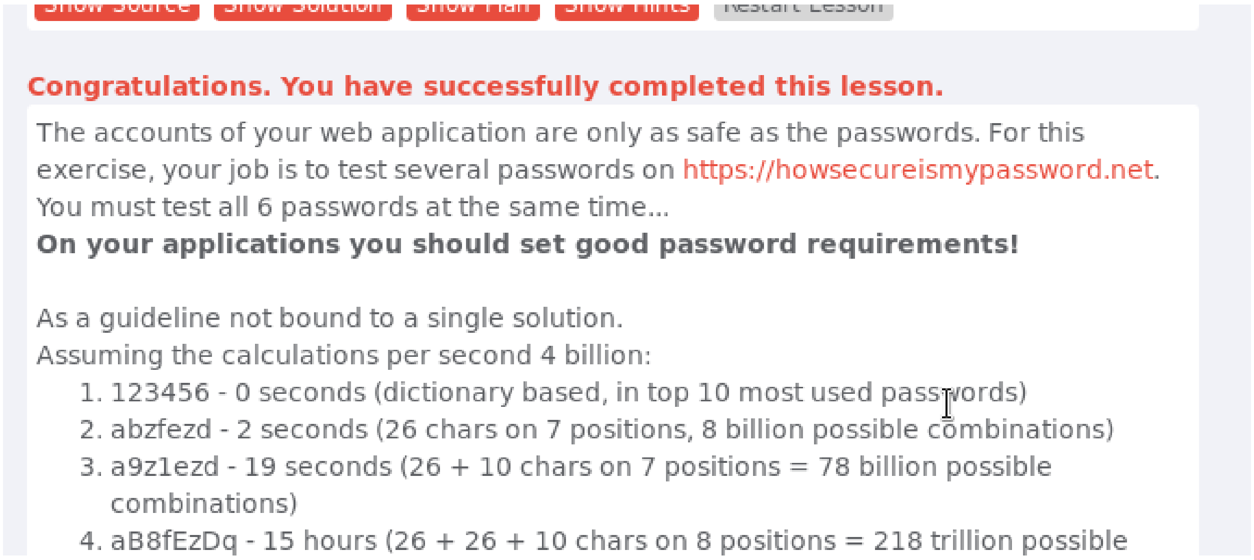
25 MICROSECONDS

to crack your password

Why not try Dashlane to create and remember stronger passwords? It's free!

TIP: USE A PASSWORD MANAGER TO SECURE AND EASILY REMEMBER YOUR PASSWORDS

Figure 16



Congratulations. You have successfully completed this lesson.

The accounts of your web application are only as safe as the passwords. For this exercise, your job is to test several passwords on https://howsecureismypassword.net. You must test all 6 passwords at the same time...
On your applications you should set good password requirements!

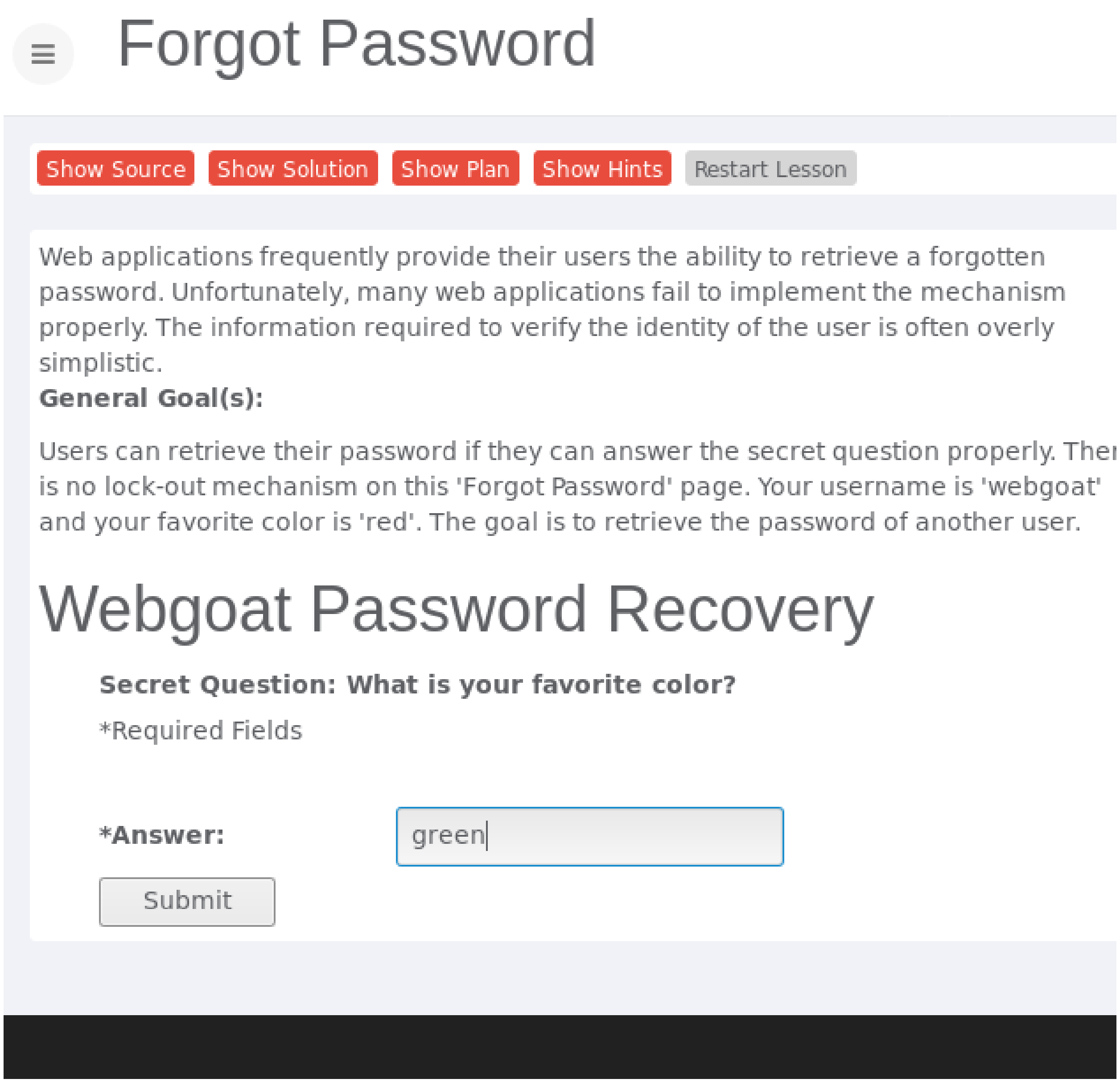
As a guideline not bound to a single solution.
Assuming the calculations per second 4 billion:

1. 123456 - 0 seconds (dictionary based, in top 10 most used passwords)
2. abzfezd - 2 seconds (26 chars on 7 positions, 8 billion possible combinations)
3. a9z1ezd - 19 seconds (26 + 10 chars on 7 positions = 78 billion possible combinations)
4. aB8fEzDq - 15 hours (26 + 26 + 10 chars on 8 positions = 218 trillion possible

Figure 17

Forgot password

- The security question, Choose the color is very weak, and you can try attempting with different color names.
- The right password was green



Forgot Password

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

General Goal(s):

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

Webgoat Password Recovery

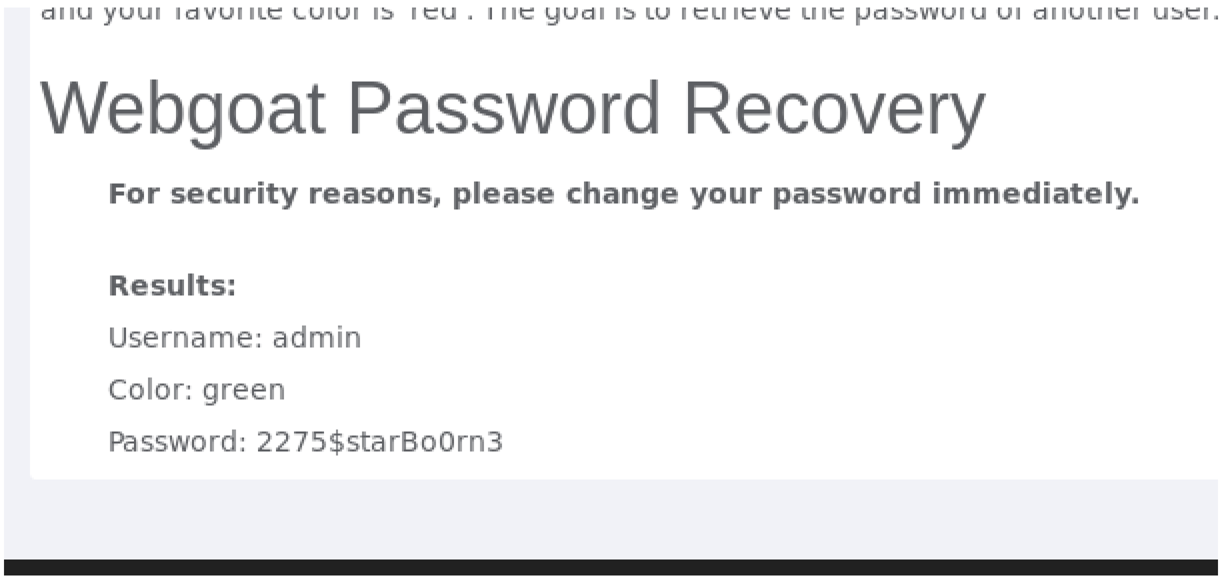
Secret Question: What is your favorite color?

*Required Fields

***Answer:** green

Submit

Figure 18


and your favorite color is 'red'. The goal is to retrieve the password of another user.

Webgoat Password Recovery

For security reasons, please change your password immediately.

Results:

Username: admin

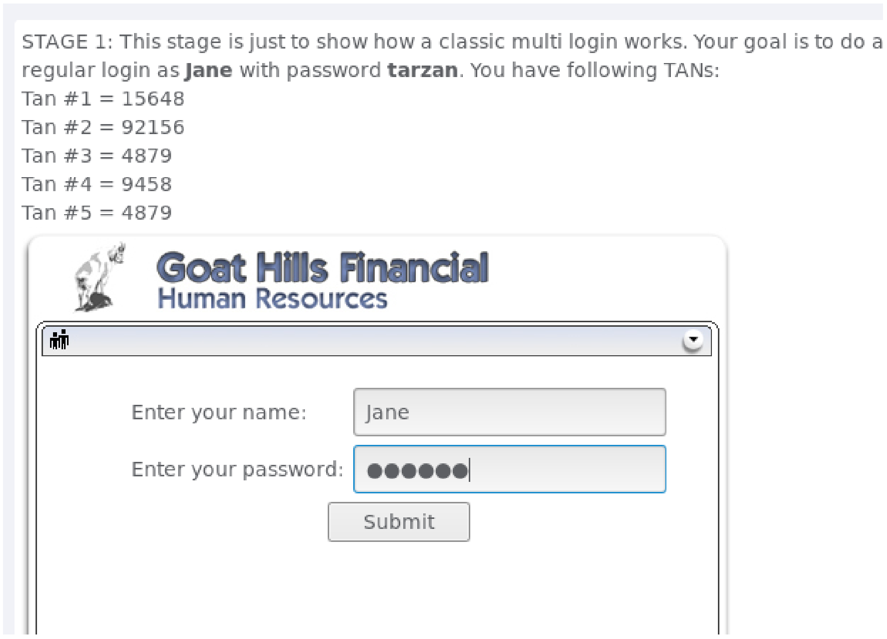
Color: green

Password: 2275$starBo0rn3

Figure 19

Multi-Level Login 1

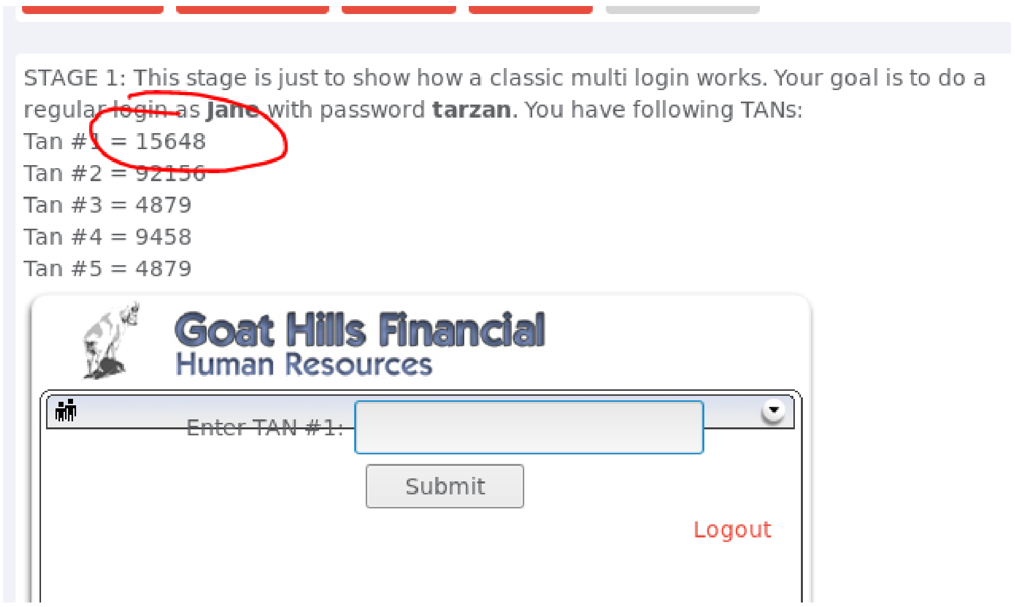
- In this part of the, we will login with the username and password given that is
  Username: Jane
  Password tarzan
- We will login using the tan#1
- Once we login, we will log out and try to login again
- Now it will ask for tan 2, in the intercept we will replace tan2 with tan1 using the tan1 id and it should log you in



STAGE 1: This stage is just to show how a classic multi login works. Your goal is to do a regular login as **Jane** with password **tarzan**. You have following TANs:
Tan #1 = 15648
Tan #2 = 92156
Tan #3 = 4879
Tan #4 = 9458
Tan #5 = 4879

Enter your name: Jane
Enter your password: ●●●●●●

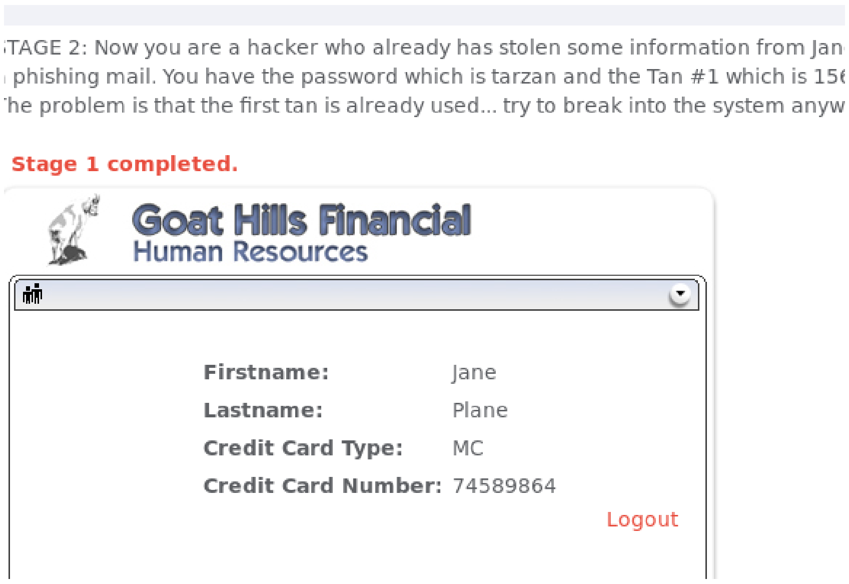
Figure 20



STAGE 1: This stage is just to show how a classic multi login works. Your goal is to do a regular login as **Jane** with password **tarzan**. You have following TANs:
Tan #1 = 15648
Tan #2 = 92156
Tan #3 = 4879
Tan #4 = 9458
Tan #5 = 4879

Enter TAN #1:

Figure 21



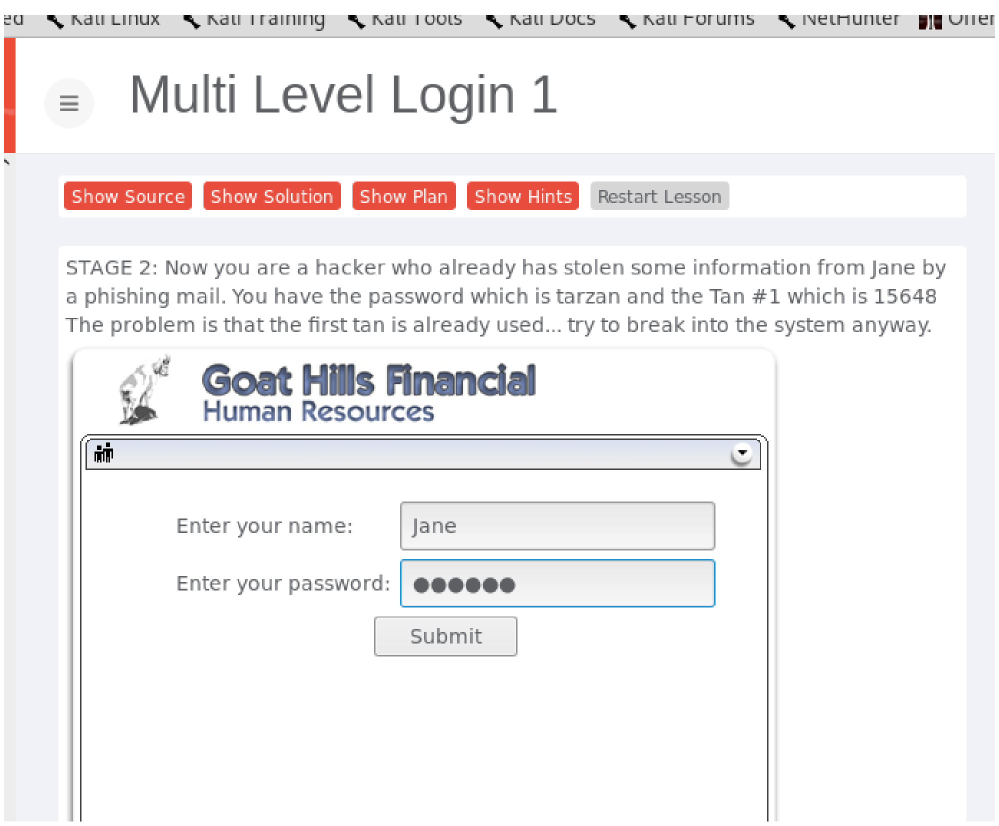
;TAGE 2: Now you are a hacker who already has stolen some information from Jan
phishing mail. You have the password which is tarzan and the Tan #1 which is 156
'he problem is that the first tan is already used... try to break into the system anyw

Stage 1 completed.

Goat Hills Financial
Human Resources

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

Figure 22



Multi Level Login 1

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

Goat Hills Financial
Human Resources

Enter your name: Jane

Enter your password: ●●●●●●

Figure 23

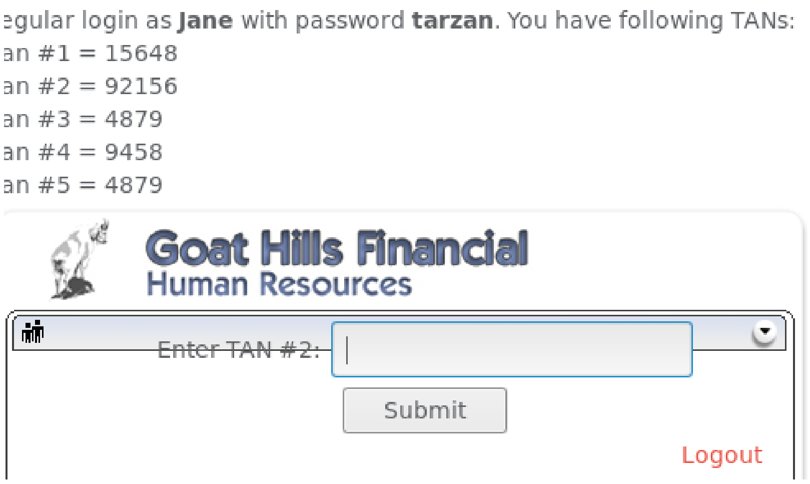

egular login as **Jane** with password **tarzan**. You have following TANs:
an #1 = 15648
an #2 = 92156
an #3 = 4879
an #4 = 9458
an #5 = 4879

Figure 24

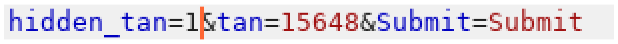
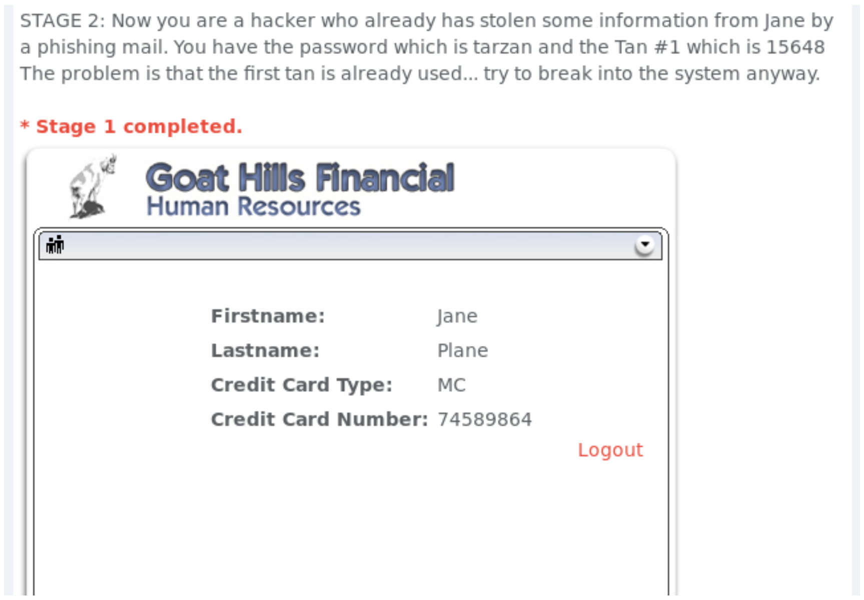

```
hidden_tan=1&tan=15648&Submit=Submit
```

Figure 25


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

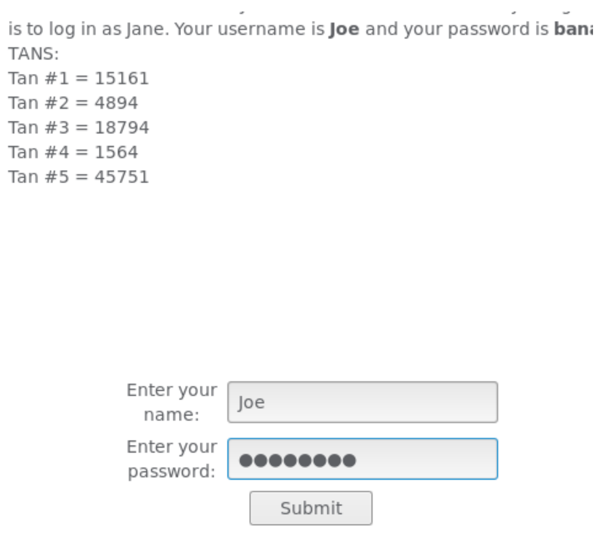
*** Stage 1 completed.**

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

Figure 26

Multi-Level login 2

- In this part, we have to log into Joe using the Jane id.
- We will enter the Joe username and bananana password.
- We will start the intercept, when it ask for tan1, we will replace the hidden_user name with Jane and forward it. It should now log into Jane's account using the Joe's ID and password.

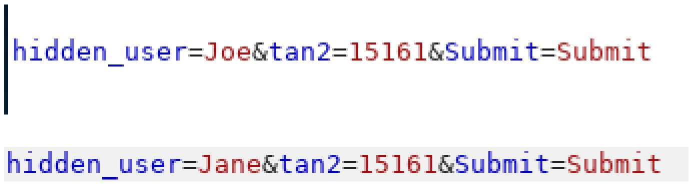


is to log in as Jane. Your username is **Joe** and your password is **ban**

TANS:
Tan #1 = 15161
Tan #2 = 4894
Tan #3 = 18794
Tan #4 = 1564
Tan #5 = 45751

Enter your name:	Joe
Enter your password:	●●●●●●●●●

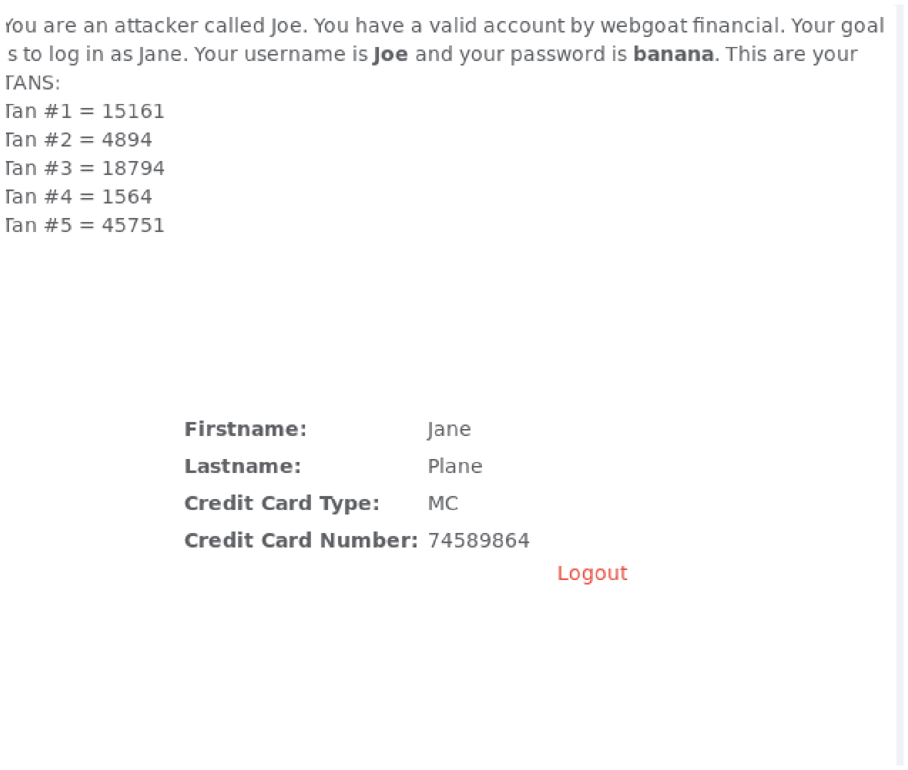
Figure 27



```
hidden_user=Joe&tan2=15161&Submit=Submit
```

```
hidden_user=Jane&tan2=15161&Submit=Submit
```

Figure 28



You are an attacker called Joe. You have a valid account by webgoat financial. Your goal s to log in as Jane. Your username is **Joe** and your password is **banana**. This are your TANS:
Tan #1 = 15161
Tan #2 = 4894
Tan #3 = 18794
Tan #4 = 1564
Tan #5 = 45751

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

Logout

Figure 29

Session Management Flaws

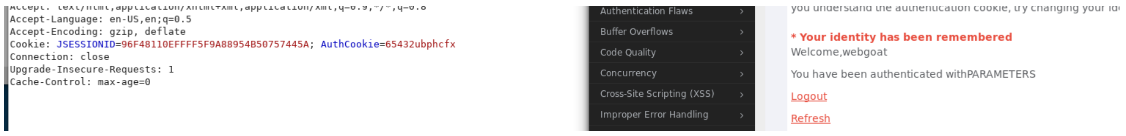
CVE-2019-1965
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-1965

This vulnerability was found in virtual shell session management for cisco NX-OS software. It was able to allow remote hacker to cause a virtual shell process to fail to delete upon termination. When there is no system memory available this could cause unexpected behavior and crashes. An attacker can exploit this vulnerability by perforing a remote management connection to the device and terminating the connection in an unexpected manner. A successful exploit could allow the attacker to cause fail to delete vsh process which can lead to a system wide denial of service.

There are no work arounds to this vulnerability. While cisco has released some software updates that addresses this vulnerability

Spoof An authentication cookie

- In this part of the lab, we will be spoofing an authentication cookie to login to a new user.
- We are given the login credentials of two users webgoat and aspect. We will log in to both account and copy the auth cookies .
- We see that both of them have auth cookie have similar numbers except last few digits.
- Cookie uses reverse-shift encryption
- We will shift the letters and then reverse it
- Webgoat → ***ubphcfx*** → shift → ***taogbew*** → reverse → ***webgoat***
- ASPECT → shift → ***udfqtb*** → ***tcepsa*** → reverse → ***aspect***
- Similarly, if we want to login to alice account the auth cookie should be
  Alice → reverse → ***ecila*** → shift → ***fdjmb***
- We will use this cookie and using intercept we will login to the account and you will see that we were logged in to the Alice's account



```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=96F48110EFFFF5F9A88954B50757445A; AuthCookie=65432ubphcfx
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

you understand the authentication cookie, try changing your id

* Your identity has been remembered

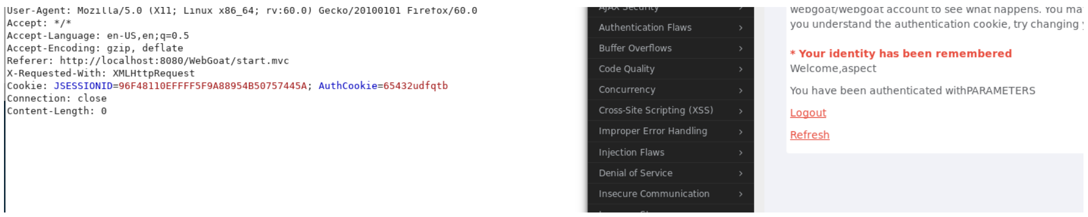
Welcome,webgoat

You have been authenticated withPARAMETERS

Logout

Refresh

Figure 30



```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=96F48110EFFFF5F9A88954B50757445A; AuthCookie=65432udfqtb
Connection: close
Content-Length: 0
```

webgoat/webgoat account to see what happens. You ma
you understand the authentication cookie, try changing

* Your identity has been remembered

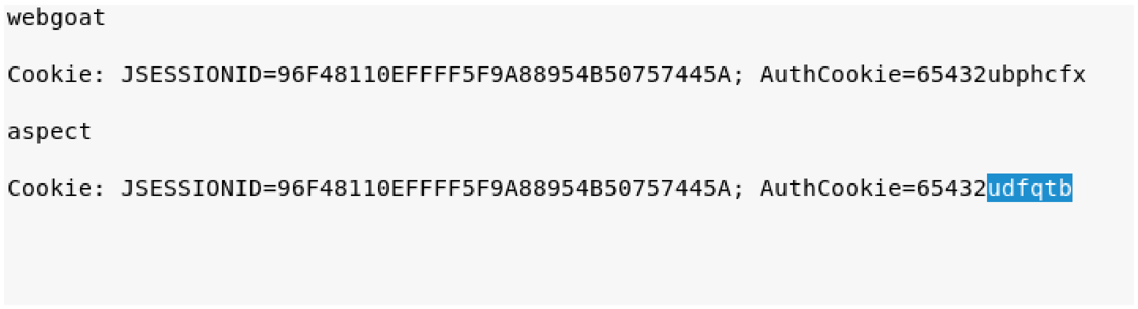
Welcome,aspect

You have been authenticated withPARAMETERS

Logout

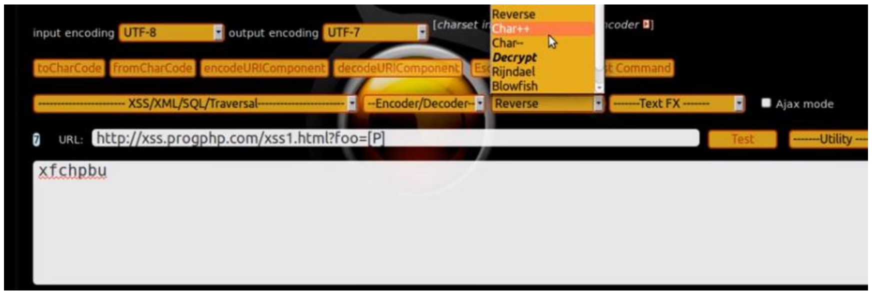
Refresh

Figure 31



```
webgoat

Cookie: JSESSIONID=96F48110EFFFF5F9A88954B50757445A; AuthCookie=65432ubphcfx

aspect

Cookie: JSESSIONID=96F48110EFFFF5F9A88954B50757445A; AuthCookie=65432udfqtb
```

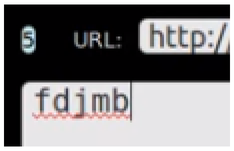
Figure 32



URL: http://xss.progphp.com/xss1.html?foo=[P]

xfchpbu

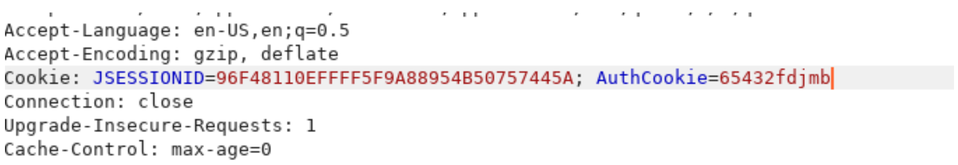
Figure 33



URL: http:/

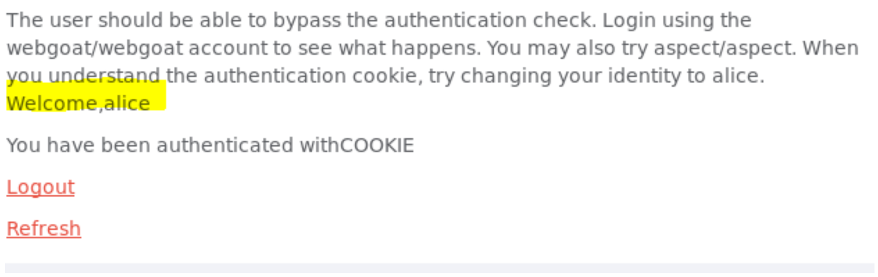
fdjmb

Figure 34



```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=96F48110EFFFF5F9A88954B50757445A; AuthCookie=65432fdjmb
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Figure 35



The user should be able to bypass the authentication check. Login using the webgoat/webgoat account to see what happens. You may also try aspect/aspect. When you understand the authentication cookie, try changing your identity to alice.

Welcome,alice

You have been authenticated withCOOKIE

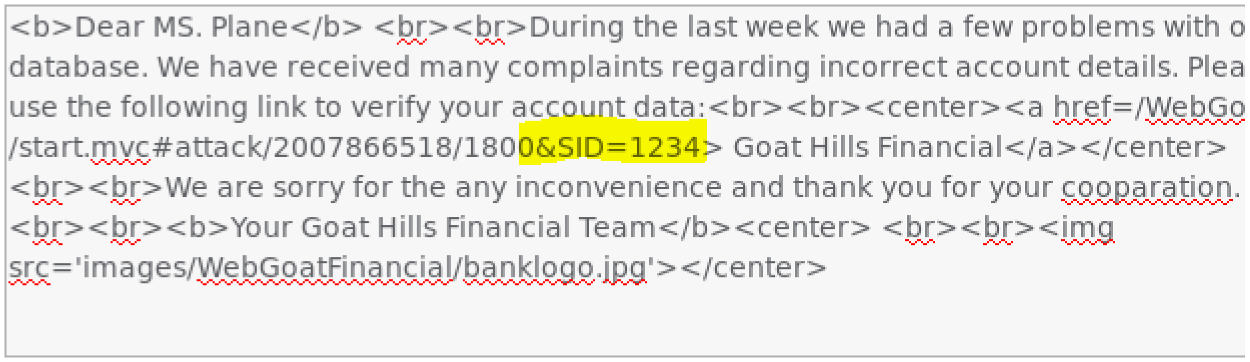
Logout

Refresh

Figure 36

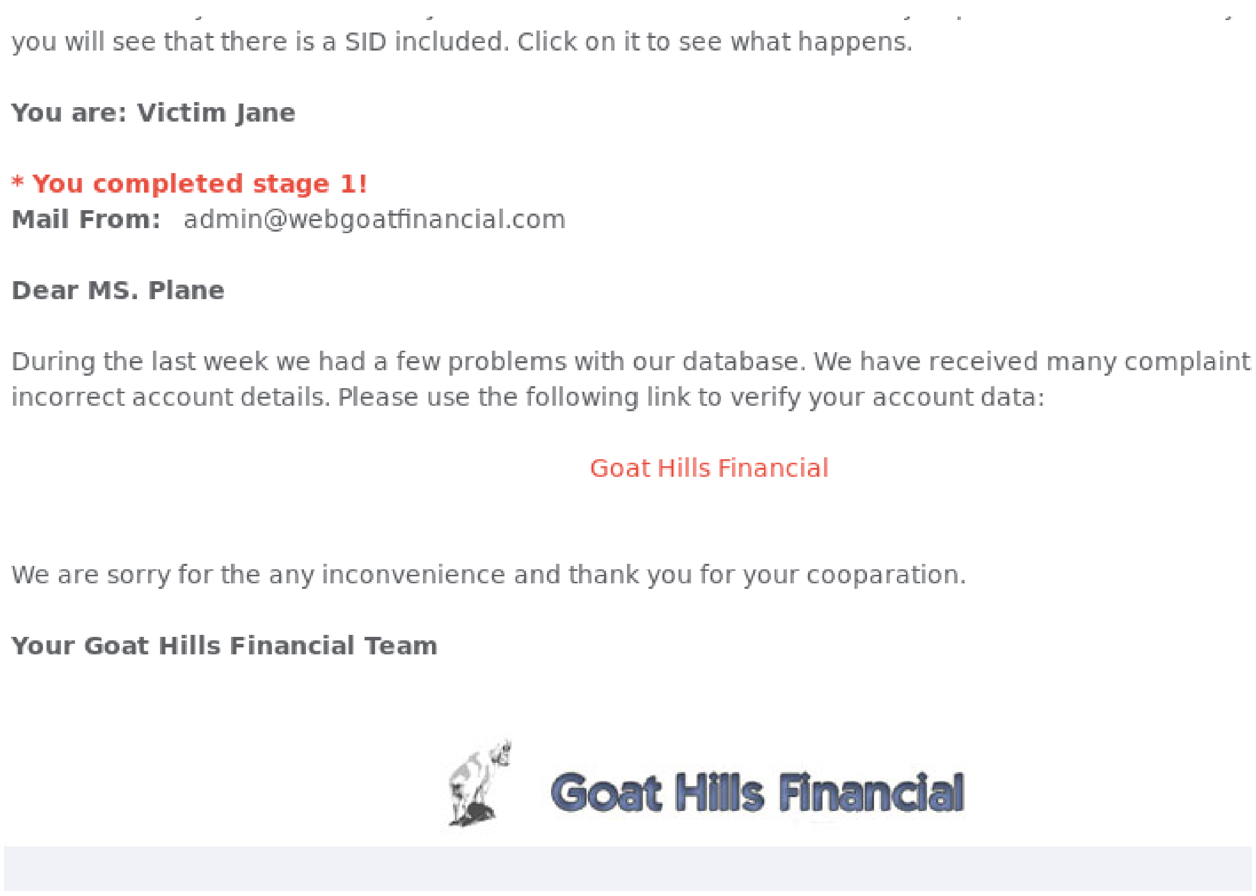
Session Fixation

- For this part of the lab, we will add a session id to the email that we want to send to the target user.
- As soon as the target opens the link in email and logs in to the link, it gets stored into the session id=
- We will now use that session id and try to login, and using that session id allows us to be logged in to that user.
- This is how we can spoof session id and login to other users.



```
<b>Dear MS. Plane</b> <br><br>During the last week we had a few problems with o
database. We have received many complaints regarding incorrect account details. Plea
use the following link to verify your account data:<br><br><center><a href=/WebGo
/start.mvc#attack/2007866518/1800&SID=1234> Goat Hills Financial</a></center>
<br><br>We are sorry for the any inconvenience and thank you for your cooparation.
<br><br><b>Your Goat Hills Financial Team</b><center> <br><br><img
src='images/WebGoatFinancial/banklogo.jpg'></center>
```

Figure 37


you will see that there is a SID included. Click on it to see what happens.

You are: Victim Jane

*** You completed stage 1!**
Mail From: admin@webgoatfinancial.com

Dear MS. Plane

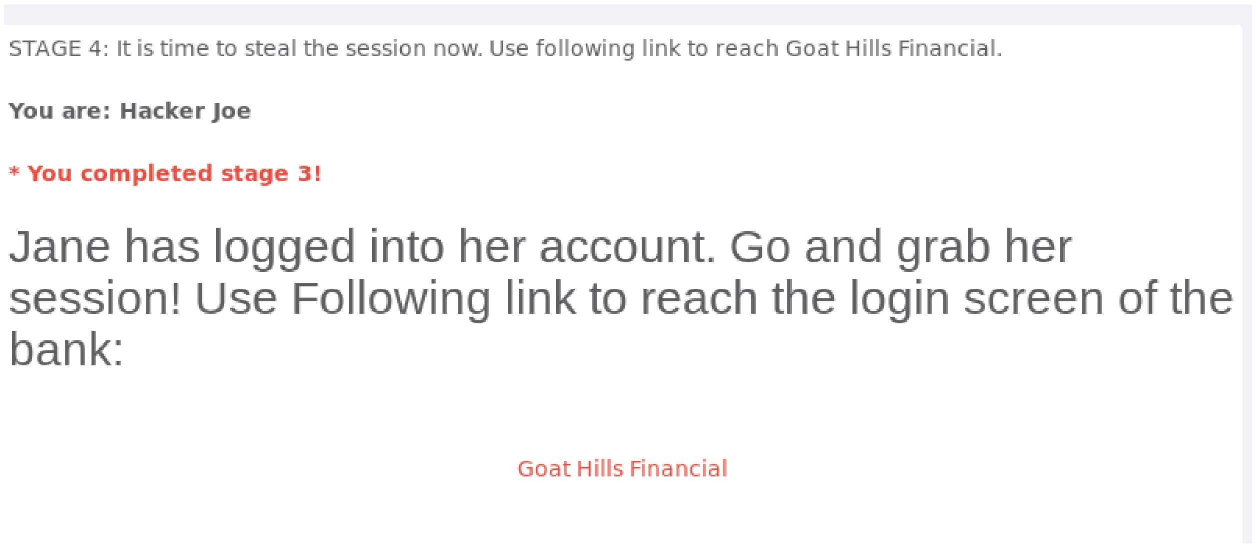
During the last week we had a few problems with our database. We have received many complaint incorrect account details. Please use the following link to verify your account data:

Goat Hills Financial

We are sorry for the any inconvenience and thank you for your cooparation.

Your Goat Hills Financial Team

Figure 38


STAGE 4: It is time to steal the session now. Use following link to reach Goat Hills Financial.

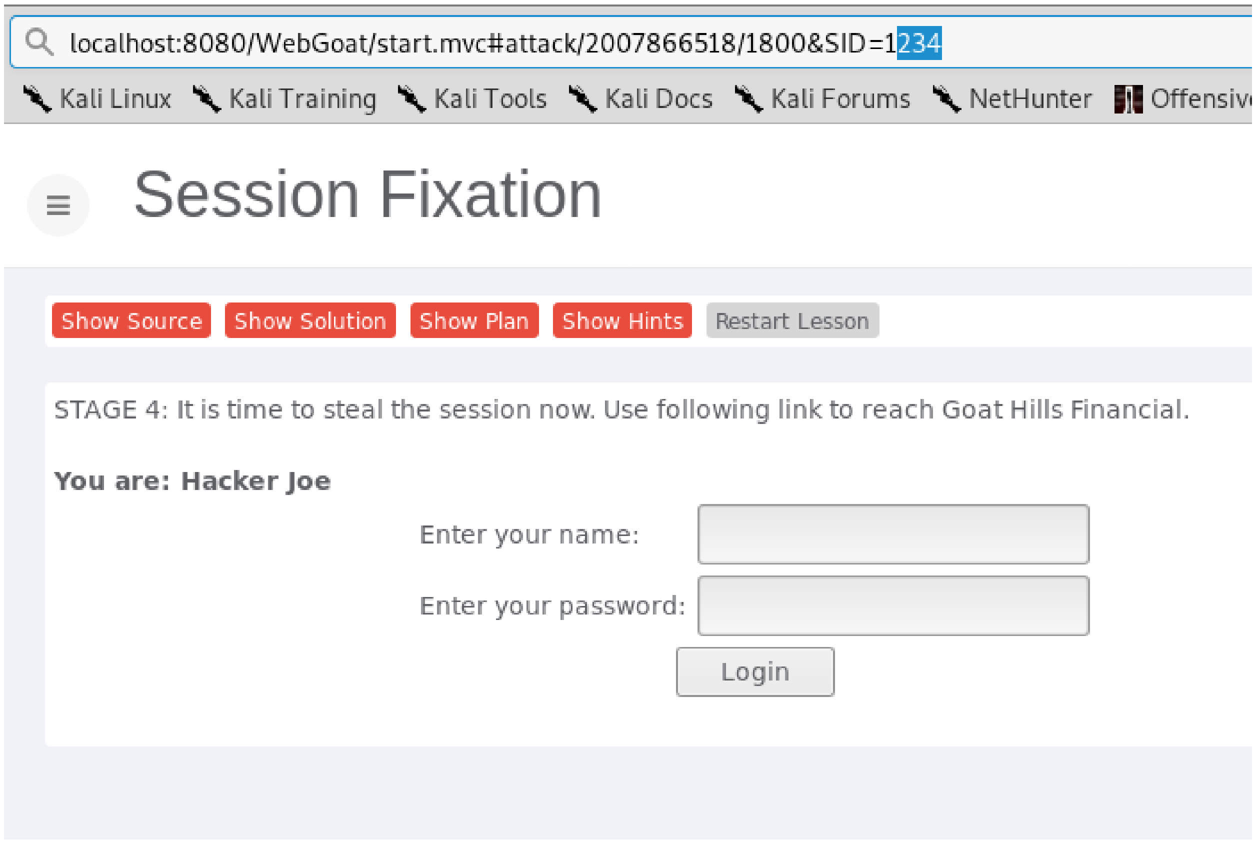
You are: Hacker Joe

*** You completed stage 3!**

Jane has logged into her account. Go and grab her session! Use Following link to reach the login screen of the bank:

Goat Hills Financial

Figure 39



Session Fixation

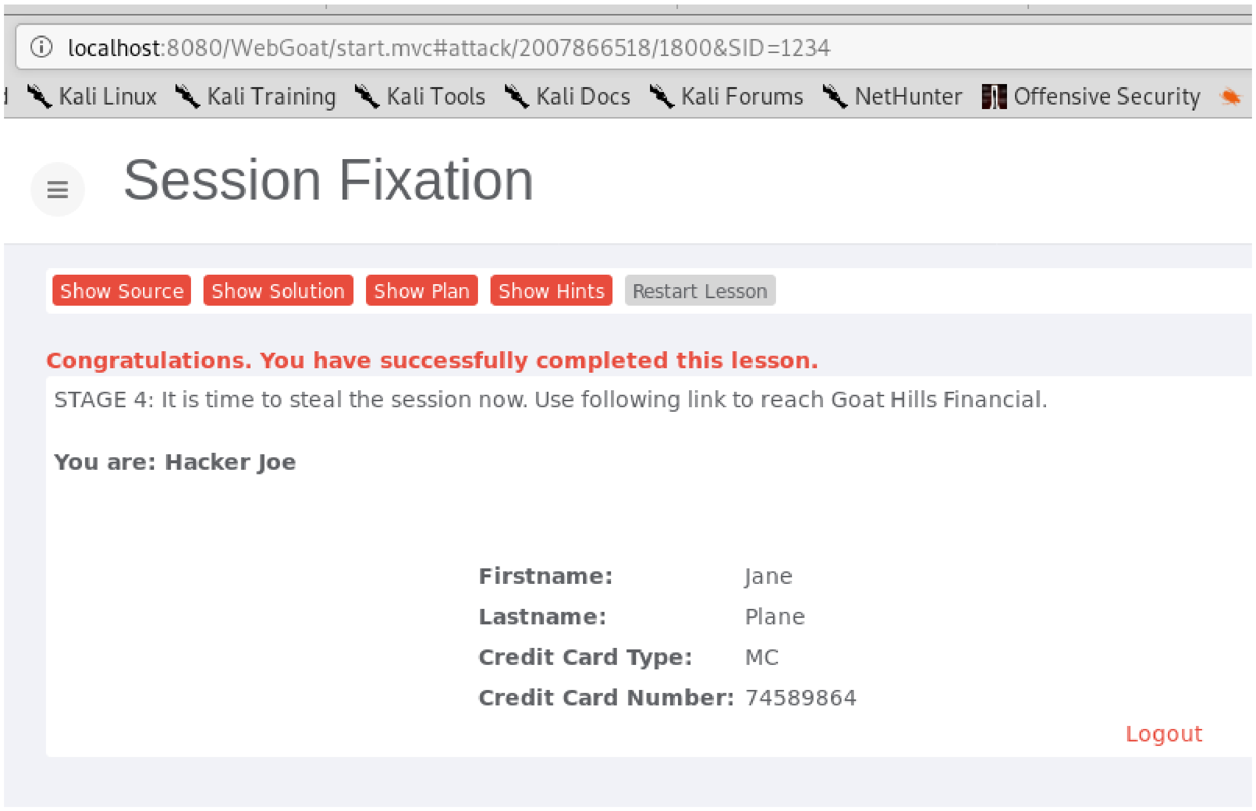
STAGE 4: It is time to steal the session now. Use following link to reach Goat Hills Financial.

You are: Hacker Joe

Enter your name:

Enter your password:

Figure 40



Session Fixation

Congratulations. You have successfully completed this lesson.

STAGE 4: It is time to steal the session now. Use following link to reach Goat Hills Financial.

You are: Hacker Joe

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

Figure 41