

BTLO — Secrets|writeup

Scenario:

You're a senior cyber security engineer and during your shift, we have intercepted/noticed a high privilege actions from unknown source that could be identified as malicious. We have got you the ticket that made these actions.

You are the one who created the secret for these tickets. Please fix this and submit the low privilege ticket so we can make sure that you deserve this position.

Here is the ticket:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmbGFnIjojQlRMe180X0V5ZXN9IiwiaWF0Ijo5MDAwMDAwMCwibmFtZSI6IkdyZWFORXhwIiwiaWF0IjoiYWRtaW4iOnRydWV9.jbkZHI  
I_W17BOALT95JQ17gIHBj9nY-oWhT1uiahtv8
```

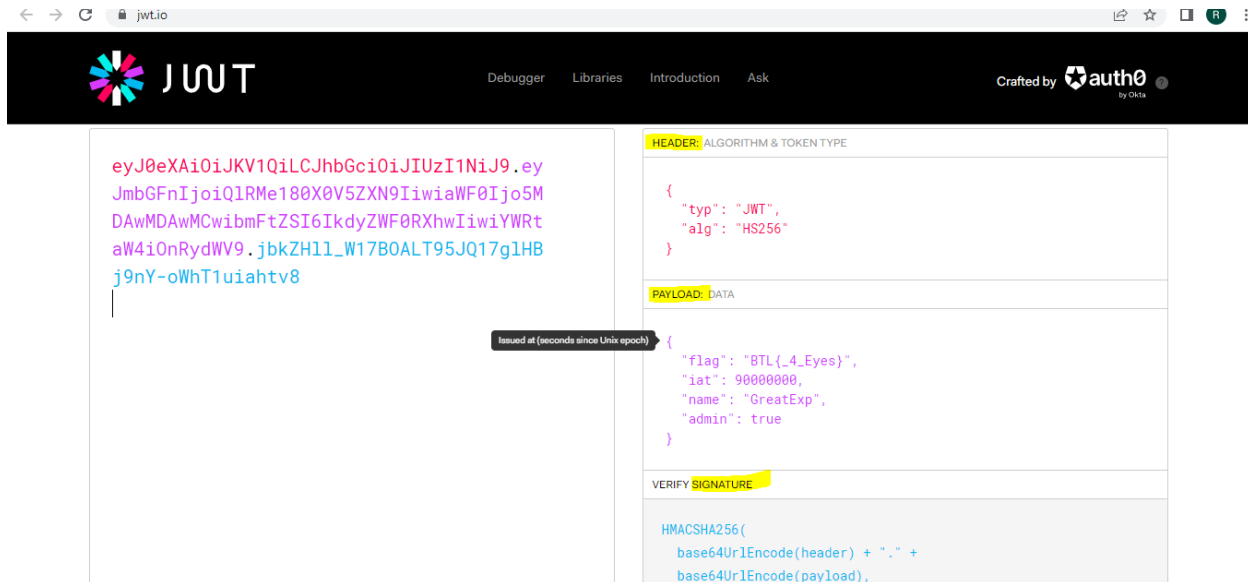
1.Can you identify the name of the token?

The given Ticket is encrypted in base64 you can decode the given token online on CyberChef since it's JWT token key.

The screenshot shows the CyberChef web application interface. On the left is the 'Operations' sidebar with a search bar and a list of operations including 'To Base64', 'From Base64', 'To Hex', 'From Hex', 'To Hexdump', 'From Hexdump', 'URL Decode', 'Regular expression', 'Entropy', 'Fork', and 'Magic'. The 'From Base64' operation is selected. The 'Recipe' panel shows the 'From Base64' operation with an 'Alphabet' dropdown set to 'A-Za-z0-9+/=' and checkboxes for 'Remove non-alphabet chars' (checked) and 'Strict mode' (unchecked). The 'Input' panel contains the Base64-encoded JWT token. The 'Output' panel shows the decoded JWT token in JSON format, with the 'typ' field highlighted in yellow. The token is: `{"typ": "JWT", "alg": "HS256"}, {"flag": "BTL{_4_Eyes}", "iat": 90000000, "name": "GreatExp", "admin": true}.1..VVx²N.ýä.51G.?gb.¡0[fj.`

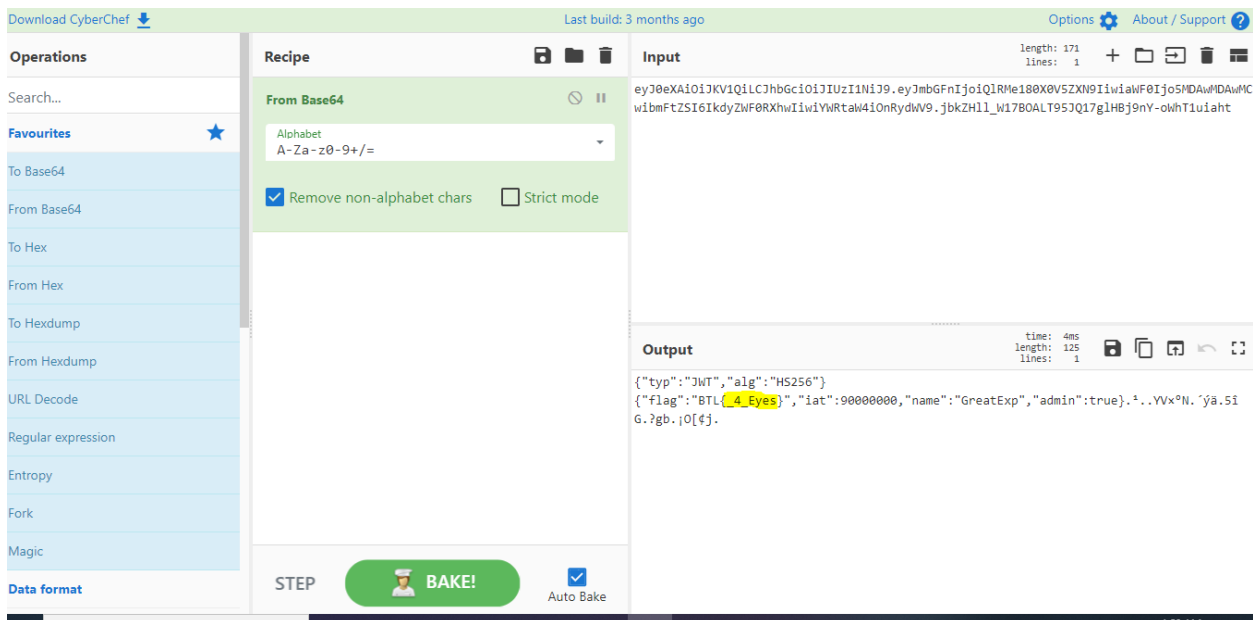
2. What is the structure of this token?

The given ticket generate a new verified signature ticket with a low privilege i generated it from <https://jwt.io/>



The screenshot shows the JWT.io website interface. On the left, a token is pasted into the input field: `eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmbGFuIjo1Q1RMe180X0V5ZXN9IiwiaWF0Ijo5MDAwMDAwMCwibmFtZSI6IkdyZWFOZXhwIiwiaWYwRtaW4iOnRydWV9.jbkZH1l_W17B0ALT95JQ17g1HBj9nY-oWhT1uiahtv8`. A tooltip indicates the token was issued at 90000000 seconds since the Unix epoch. On the right, the token is decoded into three parts: **HEADER** (ALGORITHM & TOKEN TYPE) with `{ "typ": "JWT", "alg": "HS256" }`, **PAYLOAD** (DATA) with `{ "flag": "BTL{.4_Eyes}", "iat": 90000000, "name": "GreatExp", "admin": true }`, and **VERIFY SIGNATURE** showing the HMACSHA256 algorithm and the base64 URL encoded header and payload.

3. What is the hint you found from this token?



The screenshot shows the CyberChef website interface. The 'Recipe' panel is set to 'From Base64' with the 'Remove non-alphabet chars' option checked. The 'Input' panel contains the same token as in the previous screenshot. The 'Output' panel shows the decoded token: `{ "typ": "JWT", "alg": "HS256" } { "flag": "BTL{.4_Eyes}", "iat": 90000000, "name": "GreatExp", "admin": true } .1..Vx^N. `y8.5i G.?gb.|0[4j.`. The 'Output' panel also shows the token's length (171) and lines (1).

4. What is the Secret?

To get the Secret key you have to crack the provided token so to crack the JWT token I use Hashcat. At first save the given Token in a text file, I have save it as jwt.txt.

```
(kali@kali)-[~/github_downloads]
└─$ hashcat jwt.txt -s 3 7a7a7a7a
hashcat (v6.2.5) starting in autodetect mode

OpenCL API (OpenCL 2.0 pocl 1.8 Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #1: pthread-Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz, 1347/2758 MB (512 MB allocatable), 2MCU

Hash-mode was not specified with -m. Attempting to auto-detect hash mode.
The following mode was auto-detected as the only one matching your input hash:

16500 | JWT (JSON Web Token) | Network Protocol

NOTE: Auto-detect is best effort. The correct hash-mode is NOT guaranteed!
Do NOT report auto-detect issues unless you are certain of the hash type.

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Optimizers applied:
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt
* Brute-Force

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 0 MB

Cracking performance lower than expected?

* Append -w 3 to the commandline.
  This can cause your screen to lag.

* Append -S to the commandline.
  This has a drastic speed impact but can be better for specific attacks.
  Typical scenarios are a small wordlist but a large ruleset.

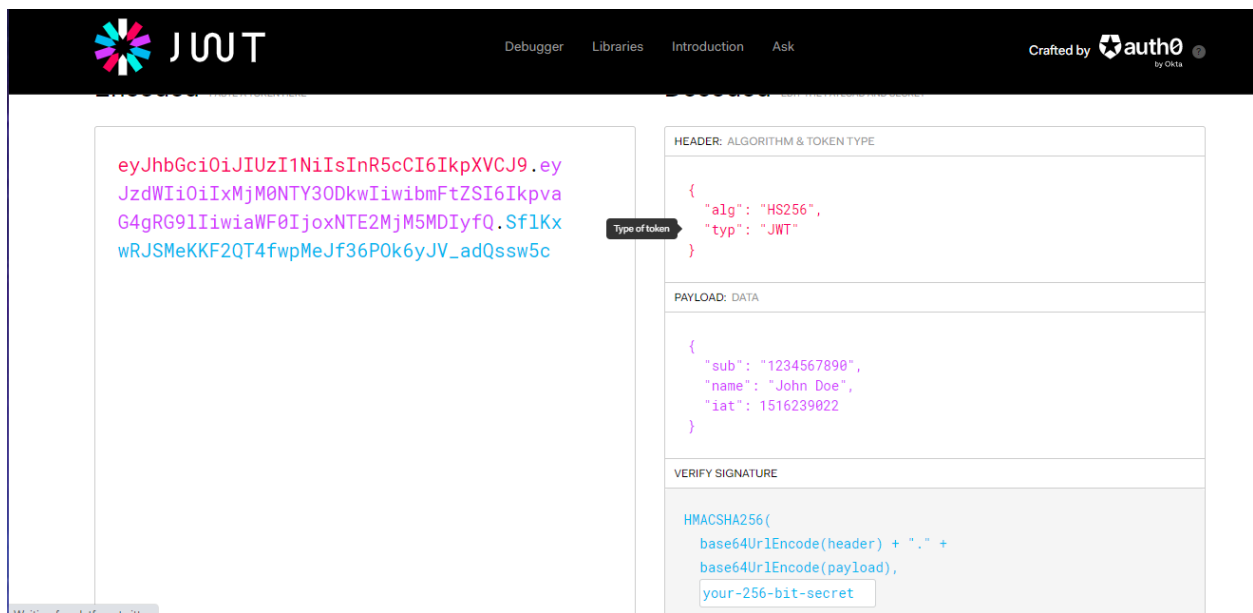
* Update your backend API runtime / driver the right way:
  https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
  https://hashcat.net/faq/morework

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmbGFuIjoiaWQlRmE1ODX0V5ZXN9IiwiaWF0Ij0iSMDAwMDAwMwibmFtZSI6IkdyZW90RkxhbmIiwiaWF0Ij0iOTYwRtaW4iOnRydWV9.fbkZhl1_W17BOALT95Jq17glHBj9nY-cWhT1uiahtv8:b7!0
```

5. Can you generate a new verified signature ticket with a low privilege?

generate a new verified signature ticket with a low privilege i generated it from <https://jwt.io/>. You have to change admin value as false and add secret key.



The screenshot shows the JWT.io website interface. On the left, a text input field contains a long alphanumeric string: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.Sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c`. A tooltip labeled "Type of token" points to the input field. On the right, the token is decoded into three parts:

- HEADER: ALGORITHM & TOKEN TYPE**:

```
{  "alg": "HS256",  "typ": "JWT"}
```
- PAYLOAD: DATA**:

```
{  "sub": "1234567890",  "name": "John Doe",  "iat": 1516239022}
```
- VERIFY SIGNATURE**:

```
HMACSHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  your-256-bit-secret)
```

At the bottom left, a status bar indicates "Waiting for platform build".