Flag jokes, WEB 200 pti

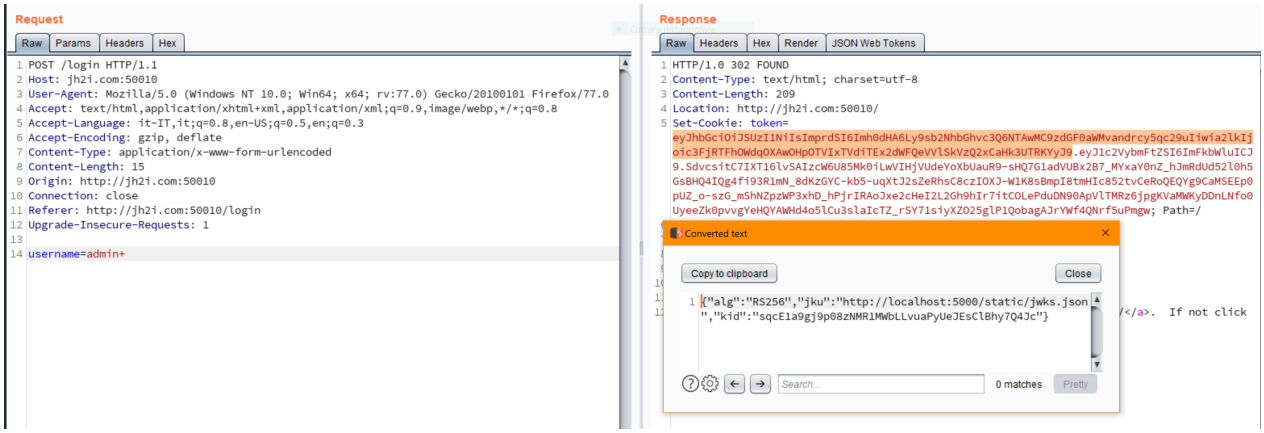
Author: p4w

TL;DR

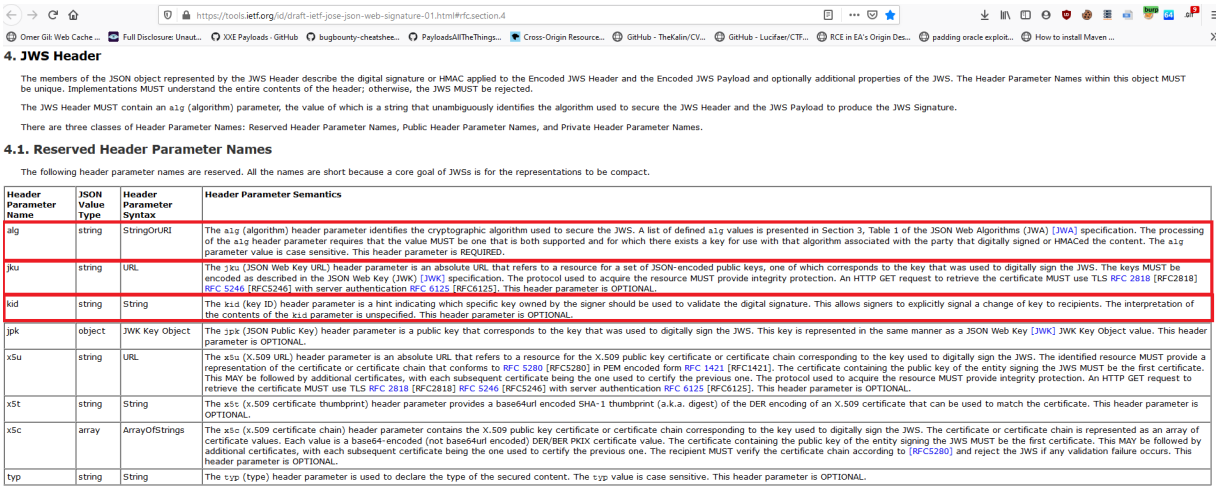
In this challenge we have to exploit a **JWKS Spoofing** vulnerability. This vulnerability exists if the application **trust an arbitrary url** to retrieve the **public key** to verify the cookie signature. An attacker can host his own public key on his malicious server in order to modify and sign the cookie with his key-pair.

Token analysis

- The challenge let us login with any username we want, except from **admin**, and the goal is to become **admin**.
- The login phase is simple we send the username we want and the application respond with a signed jwt.
- Analysis of the jwt-cookie:



- Token analysis
  - As we can notice we have 3 elements into the json part of the jwt-header and you can find the details at this link https://tools.ietf.org/id/draft-ietf-jose-json-web-signature-01.html#rfc.section.4



4. JWS Header

The members of the JSON object represented by the JWS Header describe the digital signature or HMAC applied to the Encoded JWS Header and the Encoded JWS Payload and optionally additional properties of the JWS. The Header Parameter Names within this object MUST be unique. Implementations MUST understand the entire contents of the header; otherwise, the JWS MUST be rejected.

The JWS Header MUST contain an `alg` (algorithm) parameter, the value of which is a string that unambiguously identifies the algorithm used to secure the JWS Header and the JWS Payload to produce the JWS Signature.

There are three classes of Header Parameter Names: Reserved Header Parameter Names, Public Header Parameter Names, and Private Header Parameter Names.

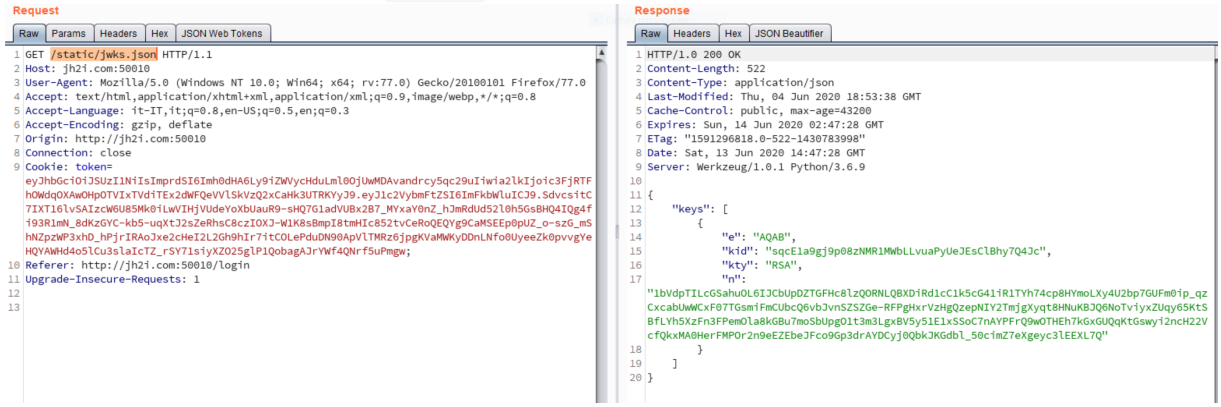
4.1. Reserved Header Parameter Names

The following header parameter names are reserved. All the names are short because a core goal of JWSs is for the representations to be compact.

Header Parameter Name	JSON Value Type	Header Parameter Syntax	Header Parameter Semantics
alg	string	StringOrURI	The `alg` (algorithm) header parameter identifies the cryptographic algorithm used to secure the JWS. A list of defined `alg` values is presented in Section 3, Table 1 of the JSON Web Algorithms (JWA) [JWA] specification. The processing of the `alg` header parameter requires that the value MUST be one that is both supported and for which there exists a key for use with that algorithm associated with the party that digitally signed or HMACed the content. The `alg` parameter value is case sensitive. This header parameter is REQUIRED.
jku	string	URL	The `jku` (JSON Web Key URL) header parameter is an absolute URL that refers to a resource for a set of JSON-encoded public keys, one of which corresponds to the key that was used to digitally sign the JWS. The keys MUST be encoded as described in the JSON Web Key (JWK) [JWK] specification. The protocol used to acquire the resource MUST provide integrity protection. An HTTP GET request to retrieve the certificate MUST use TLS RFC 2818 [RFC2818] RFC 5246 [RFC5246] with server authentication RFC 6125 [RFC6125]. This header parameter is OPTIONAL.
kid	string	String	The `kid` (key ID) header parameter is a hint indicating which specific key owned by the signer should be used to validate the digital signature. This allows signers to explicitly signal a change of key to recipients. The interpretation of the contents of the `kid` parameter is unspecified. This header parameter is OPTIONAL.
jpk	object	JWK Key Object	The `jpk` (JSON Public Key) header parameter is a public key that corresponds to the key that was used to digitally sign the JWS. This key is represented in the same manner as a JSON Web Key [JWK] JWK Key Object value. This header parameter is OPTIONAL.
x5u	string	URL	The `x5u` (X.509 URL) header parameter is an absolute URL that refers to a resource for the X.509 public key certificate or certificate chain corresponding to the key used to digitally sign the JWS. The identified resource MUST provide a representation of the certificate or certificate chain that conforms to RFC 5280 [RFC5280] in PEM encoded form RFC 1421 [RFC1421]. The certificate containing the public key of the entity signing the JWS MUST be the first certificate. This MAY be followed by additional certificates, with each subsequent certificate being the one used to certify the previous one. The protocol used to acquire the resource MUST provide integrity protection. An HTTP GET request to retrieve the certificate MUST use TLS RFC 2818 [RFC2818] RFC 5246 [RFC5246] with server authentication RFC 6125 [RFC6125]. This header parameter is OPTIONAL.
x5t	string	String	The `x5t` (x.509 certificate thumbprint) header parameter provides a base64url encoded SHA-1 thumbprint (a.k.a. digest) of the DER encoding of an X.509 certificate that can be used to match the certificate. This header parameter is OPTIONAL.
x5c	array	ArrayOfStrings	The `x5c` (x.509 certificate chain) header parameter contains the X.509 public key certificate or certificate chain corresponding to the key used to digitally sign the JWS. The certificate or certificate chain is represented as an array of certificate values. Each value is a base64-encoded (not base64url encoded) DER/BER PKIX certificate value. The certificate containing the public key of the entity signing the JWS MUST be the first certificate. This MAY be followed by additional certificates, with each subsequent certificate being the one used to certify the previous one. The recipient MUST verify the certificate chain according to [RFC5280] and reject the JWS if any validation failure occurs. This header parameter is OPTIONAL.
typ	string	String	The `typ` (type) header parameter is used to declare the type of the secured content. The `typ` value is case sensitive. This header parameter is OPTIONAL.

- To summarize, the `alg` header parameter specify the algorithm used by the web-application to sign the cookie. The `kid` header parameter is used to identify a specific key in a list of keys. The `jku` header parameter is used to point to an URL which contains a set of **json encoded public keys**.

- Let's see if we can retrieve the remote `jwks.json` file pointed by the token we have



Exploiting the vuln

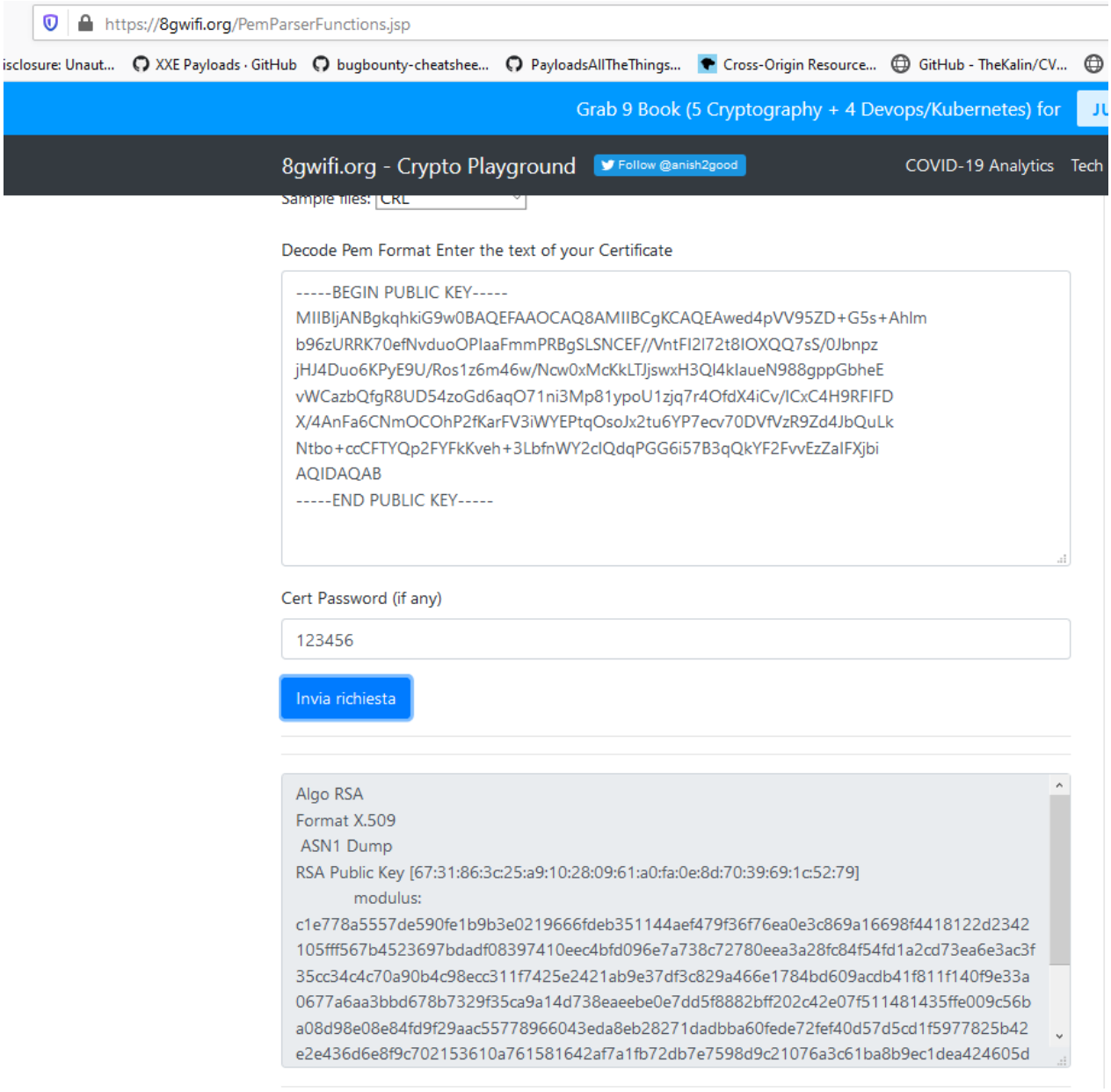
- The general idea should be to tamper the `jkw` to point to our server hosting our `jwks.json` file with our lists of keys.
- Generate RSA key-pair with `openssl`:
  - with this command we can generate the RSA private key:

```
$ openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048
```

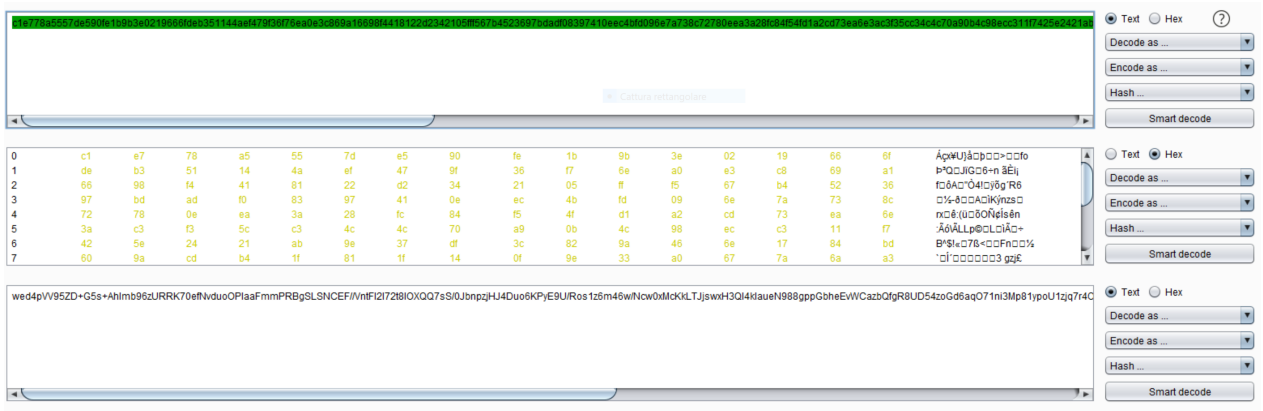
- with this command we can extract the public certificate associated to the RSA private key previously created:

```
$ openssl rsa -pubout -in private_key.pem -out public_key.pem
```

- Now we should have both **PUBLIC** and **PRIVATE** certificate in our current folder and we need to extract the `n` value. To do that I used an online service:



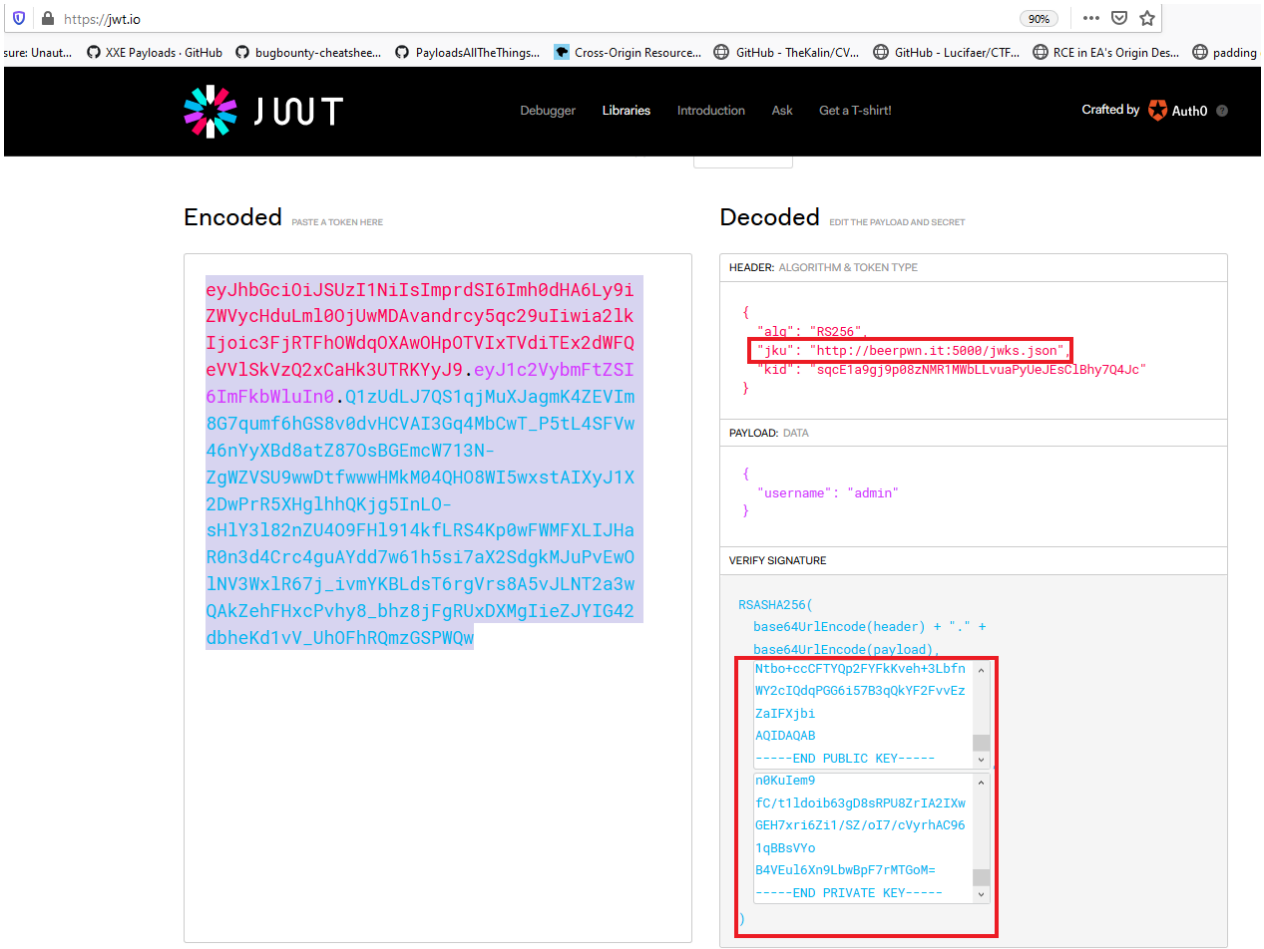
- Since the values for the public key contained in the `jwks.json` shold be in base64, then we need to convert the n from `hex` to `base64(raw-hex-bytes)`. To do that we can use burp-proxy:



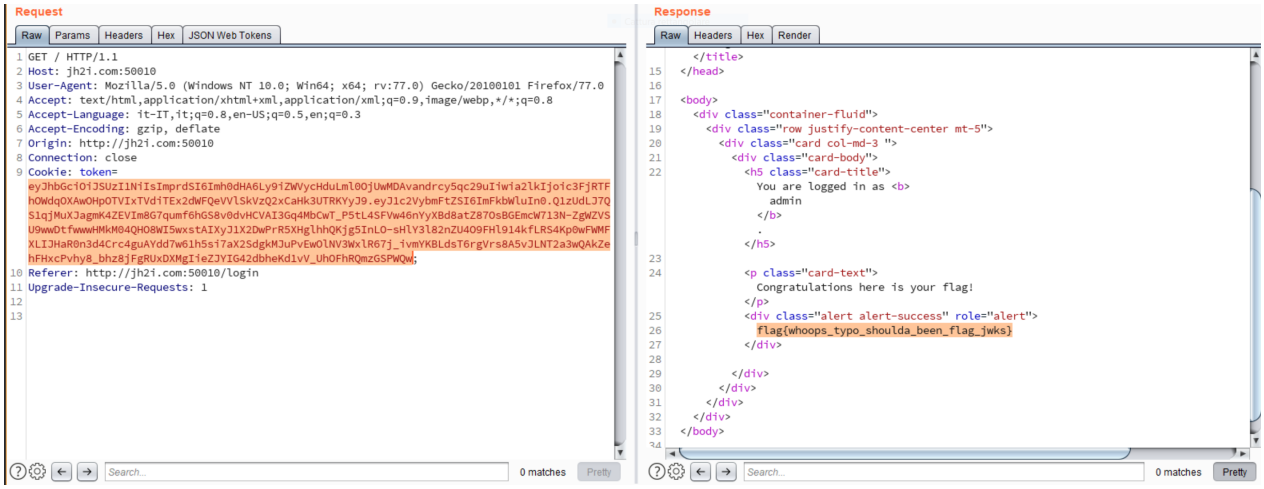
- Create our `jwks.json` file. To do that we can just use the original one and replace the value of **n** with the value associated with our RSA-key:

```
$ cat jwks.json
{
    "keys": [
        {
            "e": "AQAB",
            "kid": "sqcE1a9gj9p08zNMR1MWbLLvuaPyUeJEsClBhy7Q4Jc",
            "kty": "RSA",
            "n": "wed4pVV95ZD+G5s+Ahlmb96zURRK70efNvduoOPIaaFmmPRBgSLSNCEF//VntFI2l72t8IOXQQ7sS/0JbnpzjHJ4Duo6KPyE9U/
        }
    ]
}
```

- Now we need to modify the **jku** value of the jwt-header to point to our server where we hosted the `jwks.json` file, and modify the payload data to become `admin`. After we can sign the cookie with our hosted RSA key-pair and to do that I used the https://jwt.io service:



- Send the request to the challenge web application and get the **flag**



reference:

- https://github.com/ticarpi/jwt_tool/wiki/Known-Exploits-and-Attacks
- https://tools.ietf.org/id/draft-ietf-jose-json-web-signature-01.html#rfc.section.4
- https://book.hacktricks.xyz/pentesting-web/hacking-jwt-json-web-tokens
- https://jwt.io/