



# Les JWTs

Les erreurs à ne pas commettre.




# Les cookies


- Générés par le serveur
- Mémoriser des informations
- Stockent des informations côté-client
- Font gagner du temps
- Doivent être sécurisé

Salut c'est nous...

## les Cookies !



On a attendu d'être sûrs que le contenu de ce site vous intéresse avant de vous déranger, mais on aimerait bien vous accompagner pendant votre visite...  
C'est OK pour vous ?

Consentements certifiés par  axeptio

Non merci	Je choisis	<b>OK pour moi</b>
-----------	------------	--------------------



# Le standard JWT (RFC 7519)

- Basé sur le standard Json ({"id": "42042"}) et la base64

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gR  
G9lliwiaWF0IjoxNTE2MzI1ODQyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c

# Le Header

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

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

- 2 champs
  - typ
  - alg (RFC 7518)

"alg" Param Value	Digital Signature or MAC Algorithm	Implementation Requirements
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Recommended
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
RS512	RSASSA-PKCS1-v1_5 using SHA-512	Optional
ES256	ECDSA using P-256 and SHA-256	Recommended+
ES384	ECDSA using P-384 and SHA-384	Optional
ES512	ECDSA using P-521 and SHA-512	Optional
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC performed	Optional

# Le Payload

eyJzdWliOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MzkwMjJ9

{"sub":"1234567890","name":"John Doe","iat":1516239022}

- Champs standards (Registered claims) (rfc7519#section-4.1)
- Champs public / privé (IANA JSON Web Token Registry)
- Le champ kid (Key Identifier): "kid":"/root/keys/secret\_1.key" / "kid":"988754"

iss	Issuer
sub	Subject
aud	Audience
exp	Expiration Time
nbf	Not Before
iat	Issued at
jti	JWT ID



# Les attaques

- Les principes essentiels:

- Ne jamais faire confiance à l'utilisateur!
- Ne pas utiliser de clé de vérification faible ni par défaut
- Sécuriser son environnement (LFI / RFI / SSRF / Directory Transversal...)



# Les attaques centrées sur alg

- L'algorithme "None" (CVE-2020-15957)
- L'attaque RSA  $\leftrightarrow$  HMAC avec (CVE-2016-10555) ou sans (CVE-2017-11424) la clé publique

```
@chal.route('/rsa-or-hmac-2/authorise/<token>/')
def authorise(token):
    try:
        decoded = jwt.decode(token, PUBLIC_KEY)
    except Exception as e:
        return {"error": str(e)}

    if "admin" in decoded and decoded["admin"]:
        return {"response": f"Welcome admin, here is your flag: {FLAG}"}
    elif "username" in decoded:
        return {"response": f"Welcome {decoded['username']}"}
    else:
        return {"error": "There is something wrong with your session, goodbye"}

@chal.route('/rsa-or-hmac-2/create_session/<username>/')
def create_session(username):
    encoded = jwt.encode({'username': username, 'admin': False}, PRIVATE_KEY, algorithm='RS256')
    return {"session": encoded.decode()}
```



# Les attaques centrées sur kid

- Injection de commande système
- Injection SQL
- SSRF, directory transversal

```
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 &"  
}
```



# Conclusion

Des questions?