[redacted]

HOW TO MESS UP JSON WEB TOKENS

confoo
#confoo

~~Wekoslav Stefanovski~~ Christian Wenz
info@christianwenz.de
@chwenz

JSON Web Tokens

2015 IETF RFC 7519

Authored by Microsoft, Ping Identity, NRI

„A compact, URL-safe means of representing claims to be transferred between two parties"

Claims

- Name-value pair with information about „something"
  User name, role, validity, …
- JWT can contain arbitrary claims
- Specification defines certain claims (not all implementations care)
  iss
  sub
  aud
  exp
  nbf
  iat
  jti

Decoded

JWT (JSON Web Token)

```
eyJhbGciOiJIUzI1NiIsInR
5cCI6IkpXVCJ9.eyJzdWIiO
iJhbGljZUBleGFtcGxlLmNv
bSIsIm5hbWUiOiJBbGljZSI
sImF1ZCI6Imh0dHBzOi8vZX
hhbXBsZS5jb20iLCJpYXQiO
jE2NDMyMzkwMjJ9.K_f6k7N
nrpamUXbDdN6BeP9HVmbumM
MLiNKoV35fVAw
```

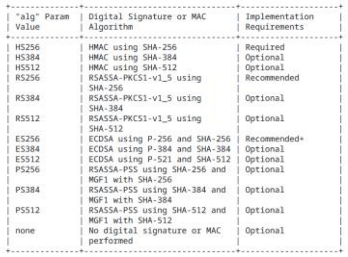
```
{
  "alg": "HS256",
  "typ": "JWT"
}
{
  "sub": "alice@example.com",
  "name": "Alice",
  "aud": "https://example.com",
  "iat": 1643239022
}
/* Signature */
```

JWT Algorithms

Specification only mentions two algorithms
HS256 (HMAC SHA-256)
none (no signature)

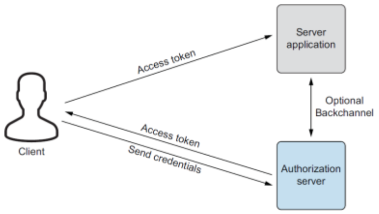
Other algorithms are defined in IETF RFC 7518
(https://datatracker.ietf.org/doc/html/rfc7518)

JWT Algorithms (2)



```
+--------------+-------------------------------+--------------------+
| "alg" Param  | Digital Signature or MAC      | Implementation     |
| Value        | Algorithm                     | 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       | Recommended        |
|              | SHA-256                       |                    |
| RS384        | RSASSA-PKCS1-v1_5 using       | Optional           |
|              | SHA-384                       |                    |
| RS512        | RSASSA-PKCS1-v1_5 using       | Optional           |
|              | SHA-512                       |                    |
| 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  | Optional           |
|              | MGF1 with SHA-256             |                    |
| PS384        | RSASSA-PSS using SHA-384 and  | Optional           |
|              | MGF1 with SHA-384             |                    |
| PS512        | RSASSA-PSS using SHA-512 and  | Optional           |
|              | MGF1 with SHA-512             |                    |
| none         | No digital signature or MAC   | Optional           |
|              | performed                     |                    |
+--------------+-------------------------------+--------------------+
```

Using JWTs



Takeaways

Tokens are (usually) not encrypted, but signed

Signing can use a shared key, or a public/private key pair

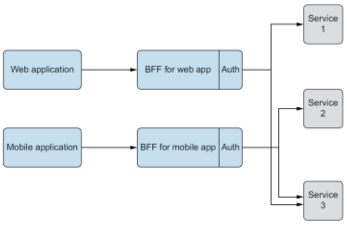
Tokens have an expiry date, but cannot be invalidated

Tokens are meant for certain systems („audiences“)

Token Attacks

- Unsigned tokens
- Self-signed tokens
- Reused tokens
- Insecure token storage

BFF Instead of Tokens in Local Storage



Further Issues

- Sensitive cleartext information in tokens
- Bloated tokens
- Ignoring token audience
- Expiration date ignored, or too far in the future

Thank You!

- Questions?
- info@christianwenz.de
- Twitter: @chwenz

https://is.gd/MIhxaZ