Applied Crypto Project

JSON Web Tokens

ECDSA, RFC 7519 (May 2015)

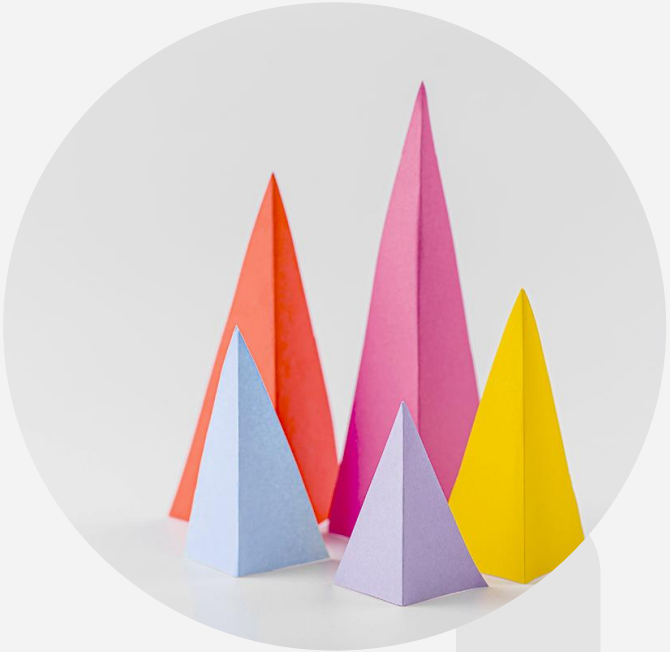
Presenter: Emin Muhammadi

JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.

Benefits

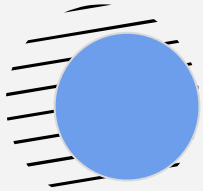
There are benefits to using JWTs when compared to simple web tokens (SWTs) and Security Assertion Markup Language (SAML) tokens.

- More compact
- More secure
- More common
- Easier to process

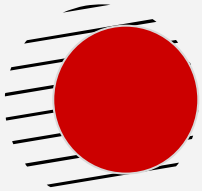


Use cases

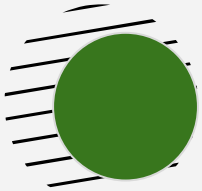
JWTs can be used in various ways:



Authentication



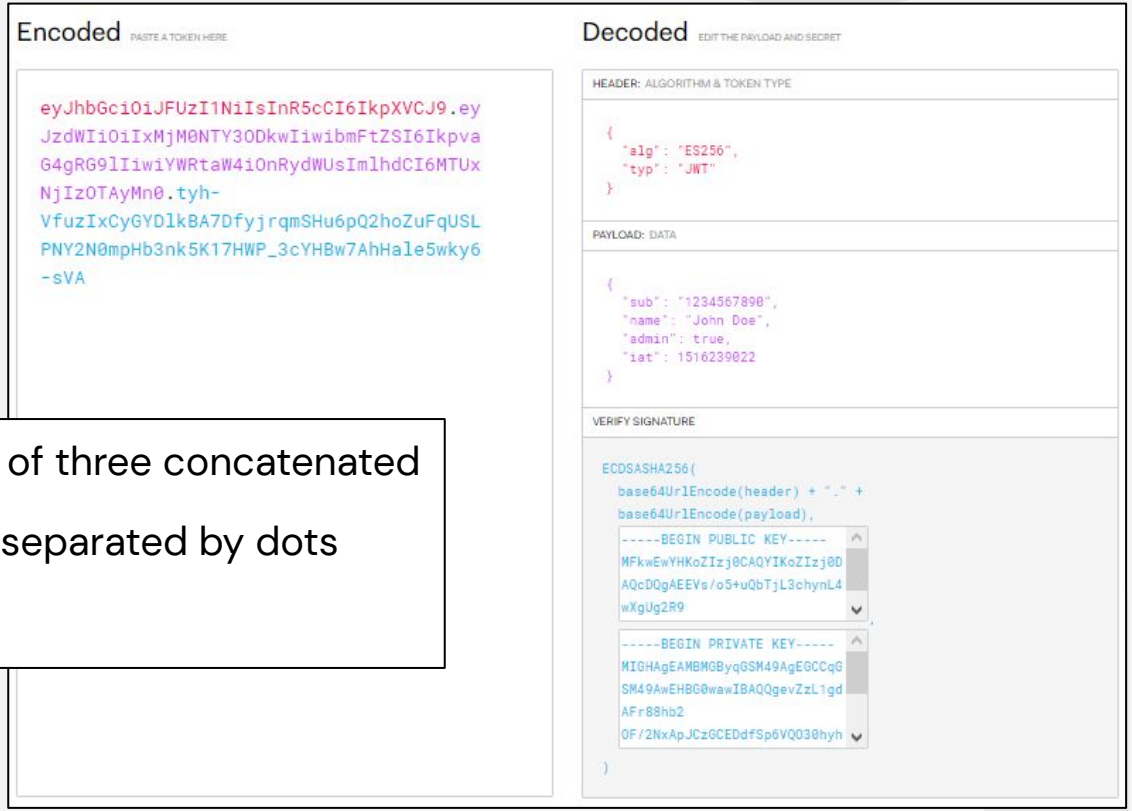
Authorization



Information Exchange

JSON Web Token Structure

- **JOSE* header**
- **JWS payload**
- **JWS signature**



A well-formed JWT consists of three concatenated Base64url-encoded strings, separated by dots

header.payload.signature

* JOSE - JSON Object Signing and Encryption

Algorithms

Most JWTs in the wild are just signed. The most common algorithms are:

- HMAC + SHA256
- RSASSA-PKCS1-v1_5 + SHA256
- ECDSA + P-256 + SHA256

JSON Web Token Claims

The JWT specification defines seven reserved claims that are not required, but are recommended to allow interoperability with third-party applications. These are:

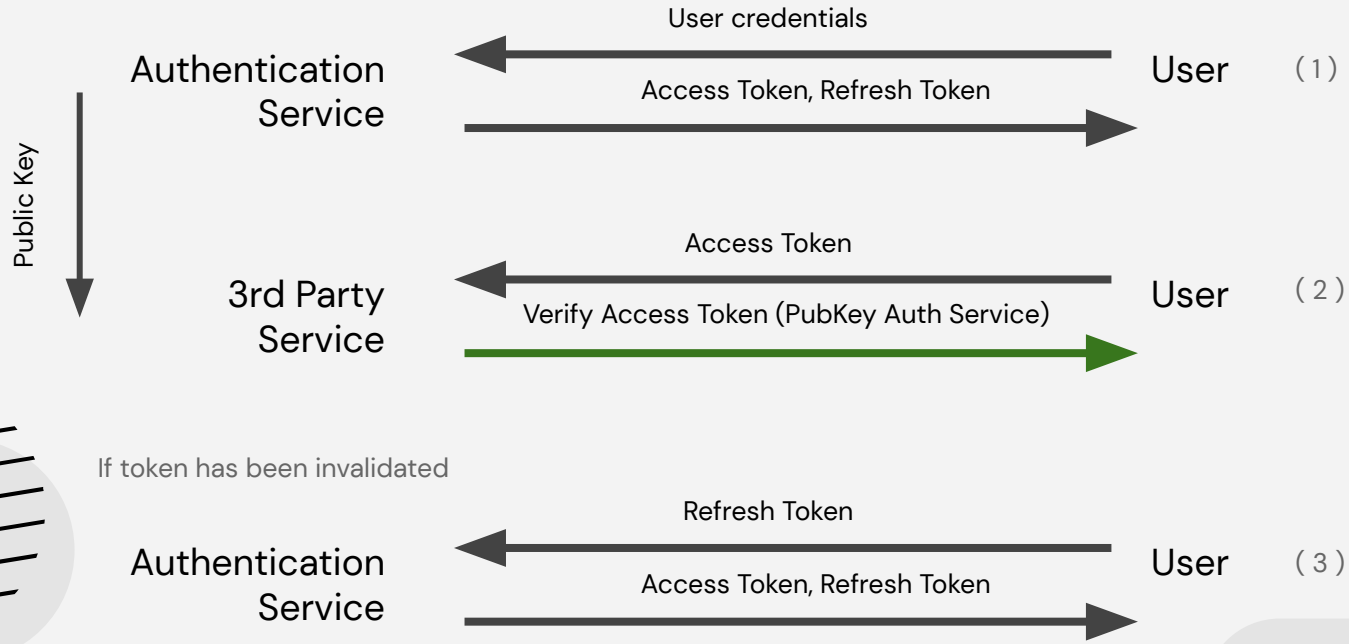
- **iss** (issuer): Issuer of the JWT
- **sub** (subject): Subject of the JWT (the user)
- **aud** (audience): Recipient for which the JWT is intended
- **exp** (expiration time): Time after which the JWT expires
- **nbf** (not before time): Time before which the JWT must not be accepted for processing
- **iat** (issued at time): Time at which the JWT was issued; can be used to determine age of the JWT
- **jti** (JWT ID): Unique identifier; can be used to prevent the JWT from being replayed (allows a token to be used only once)

Signature

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that. For example if you want to use the HMAC SHA256 algorithm, the signature will be created in the following way:

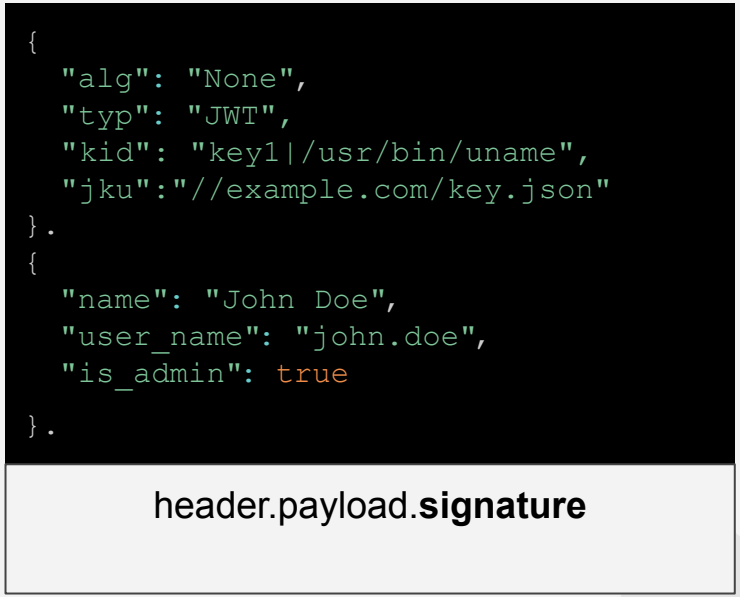
```
HMACSHA256(base64UrlEncode(header) + "." +
    base64UrlEncode(payload), secret)
```

How it works?


User credentials
Authentication Service
User
(1)
Access Token, Refresh Token
Public Key
Access Token
3rd Party Service
User
(2)
Verify Access Token (PubKey Auth Service)
If token has been invalidated
Refresh Token
Authentication Service
User
(3)
Access Token, Refresh Token
Repeat step 2.

Vulnerabilities

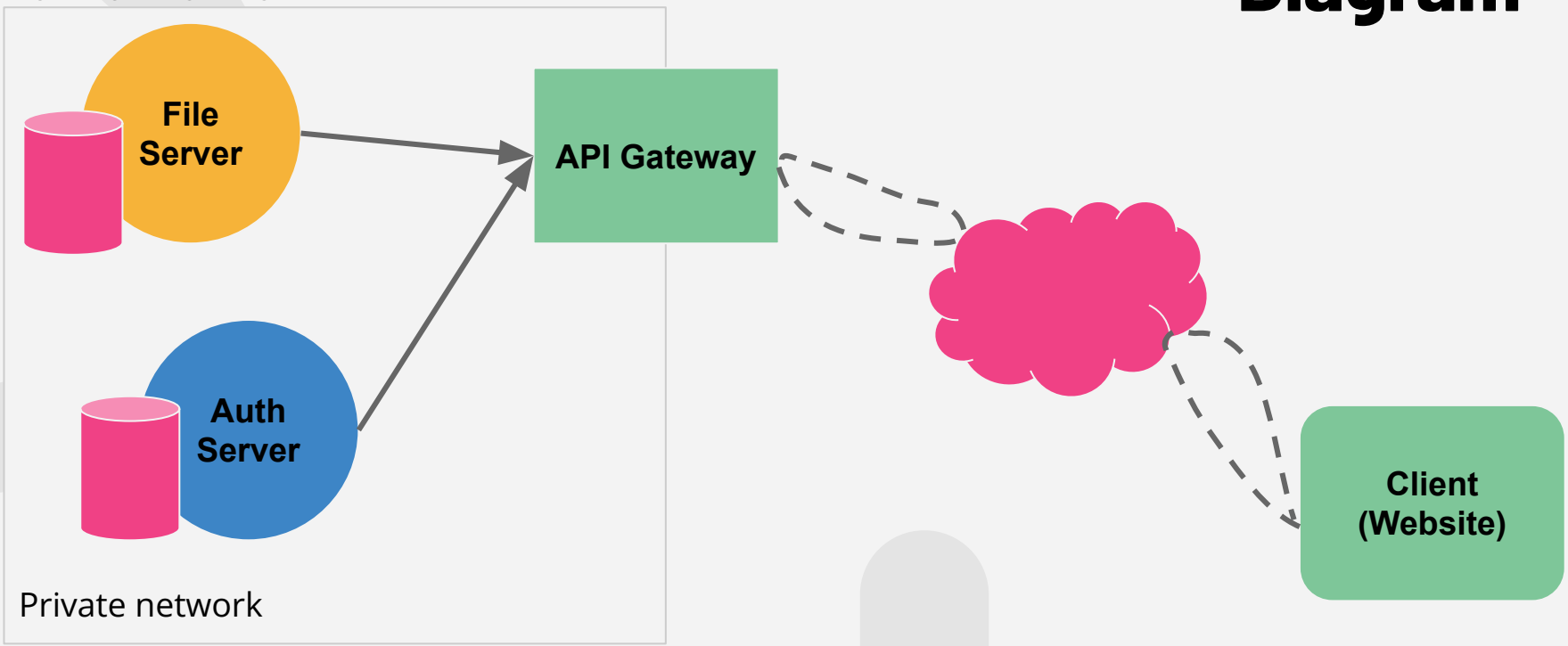
- Disable signing - None Algorithm
- Algorithm confusion - RSA to HMAC
- `kid` parameter injection
- Attacks using the `jku` header



Demo Time

https://github.com/eminmuhammadi/applied-crypto-project

Diagram


File Server
Auth Server
Private network
API Gateway
Client (Website)

Load test

```
     http_req_blocked...............: avg=1.23ms   min=0s      med=998.9µs max=35.72ms p(90)=2ms
p(95)=3.27ms
     http_req_connecting............: avg=1.11ms   min=0s      med=754.6µs max=35.72ms p(90)=1.85ms   p(95)=3ms
   ✓ http_req_duration..............: avg=4.7s     min=84.2ms  med=1.69s   max=12.04s  p(90)=11.32s
p(95)=11.54s
       { expected_response:true }...: avg=4.7s     min=84.2ms  med=1.69s   max=12.04s  p(90)=11.32s
p(95)=11.54s
     http_req_failed................: 0.00%  ✓ 0         ✗ 3526
     http_req_receiving.............: avg=216.06µs min=0s      med=0s      max=13.36ms p(90)=507.15µs
p(95)=859.65µs
     http_req_sending...............: avg=94.34µs  min=0s      med=0s      max=5.5ms   p(90)=503.7µs
p(95)=998.77µs
     http_req_tls_handshaking.......: avg=0s       min=0s      med=0s      max=0s      p(90)=0s       p(95)=0s
     http_req_waiting...............: avg=4.7s     min=83.83ms med=1.69s   max=12.04s  p(90)=11.32s
p(95)=11.54s
     http_reqs......................: 3526    14.676317/s
     iteration_duration.............: avg=10.42s   min=1.27s   med=12.97s  max=14.18s  p(90)=13.36s
p(95)=13.42s
     iterations.....................: 1763    7.338158/s
     vus............................: 2       min=2       max=100
     vus_max........................: 100     min=100     max=100
```

Smoke test

```
     http_req_blocked...............: avg=579.39µs min=0s      med=532.2µs  max=1.95ms   p(90)=786.11µs
p(95)=1.02ms
     http_req_connecting............: avg=528.67µs min=0s      med=529.05µs max=1.95ms   p(90)=648.44µs
p(95)=777.8µs
   ✓ http_req_duration..............: avg=144.38ms min=78.44ms med=179.99ms max=311.58ms p(90)=204.47ms
p(95)=209.12ms
       { expected_response:true }...: avg=144.38ms min=78.44ms med=179.99ms max=311.58ms p(90)=204.47ms
p(95)=209.12ms
     http_req_failed................: 0.00%  ✓ 0        ✗ 370
     http_req_receiving.............: avg=121.83µs min=0s      med=0s       max=1.5ms    p(90)=519.71µs
p(95)=654.85µs
     http_req_sending...............: avg=47.48µs  min=0s      med=0s       max=997.3µs  p(90)=50.55µs
p(95)=504.83µs
     http_req_tls_handshaking.......: avg=0s       min=0s      med=0s       max=0s       p(90)=0s       p(95)=0s
     http_req_waiting...............: avg=144.21ms min=78.44ms med=179.75ms max=311.58ms p(90)=204.3ms
p(95)=208.58ms
     http_reqs......................: 370     1.540924/s
     iteration_duration.............: avg=1.29s    min=1.26s   med=1.29s    max=1.42s    p(90)=1.31s
p(95)=1.32s
     iterations.....................: 185     0.770462/s
     vus............................: 1       min=1      max=1
     vus_max........................: 1       min=1      max=1
```

Thanks for your attention

Q&A

References

- Auth0 Docs - https://auth0.com/docs/
- RFC 7523 - https://datatracker.ietf.org/doc/html/rfc7523
- JSON Web Token attacks and vulnerabilities - https://www.netsparker.com/blog/web-security/json-web-token-jwt-attacks-vulnerabilities/