Attacking and Securing JWT

By @airman604 for @OWASPVanouver

$ whoami

JWT

JWT = JSON Web Tokens

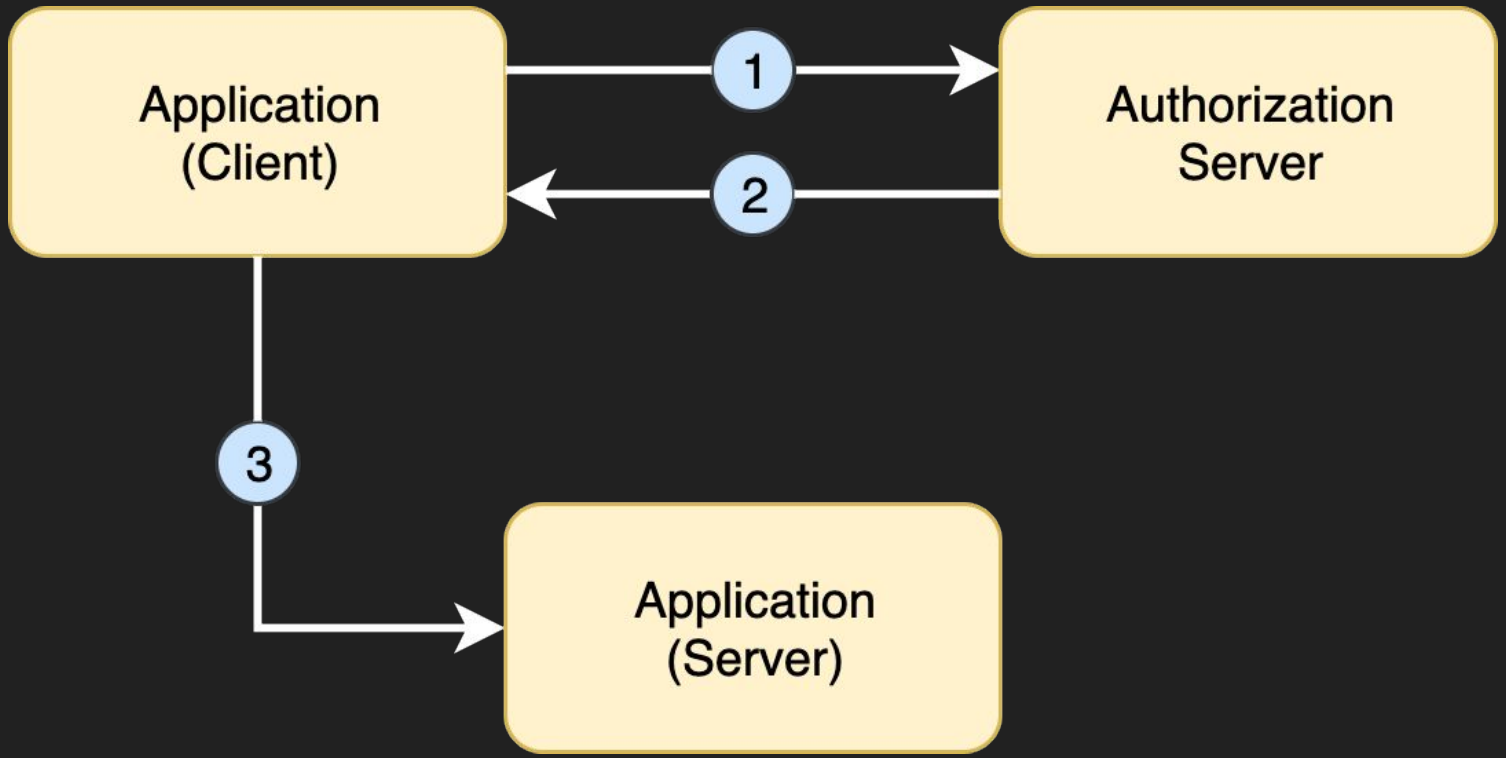
Defined in RFC 7519

Extensively used on the web, for example in OpenID Connect

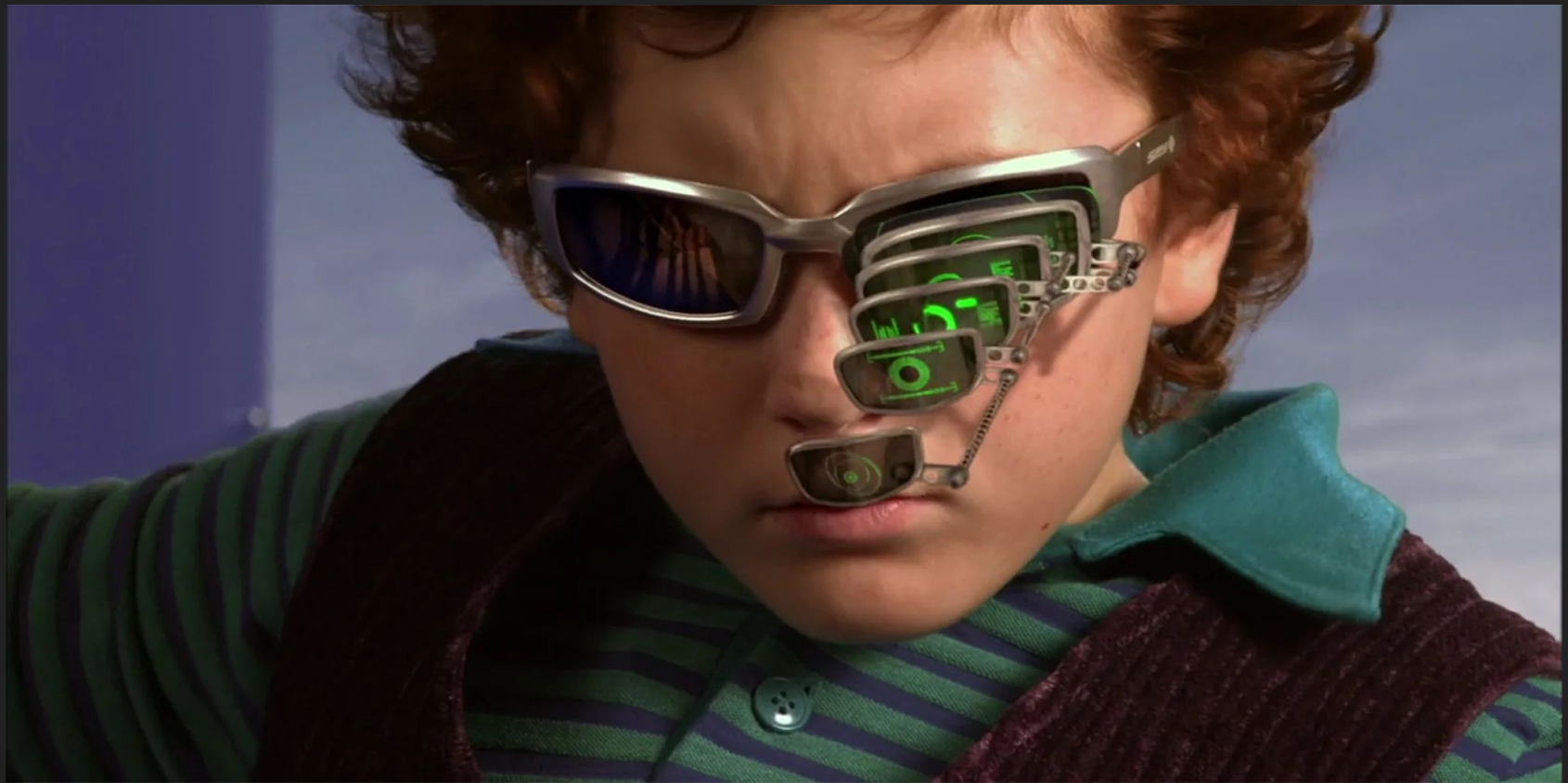
Why people use JWT?

- (Somewhat) secure way to exchange authentication information (“claims”)
- Stateless session management, no session cookies
- Once configured (establishes trust), backend doesn’t need to talk to authorization server

Typical Use


Application
(Client)
1
2
Authorization
Server
3
Application
(Server)

A Closer Look...



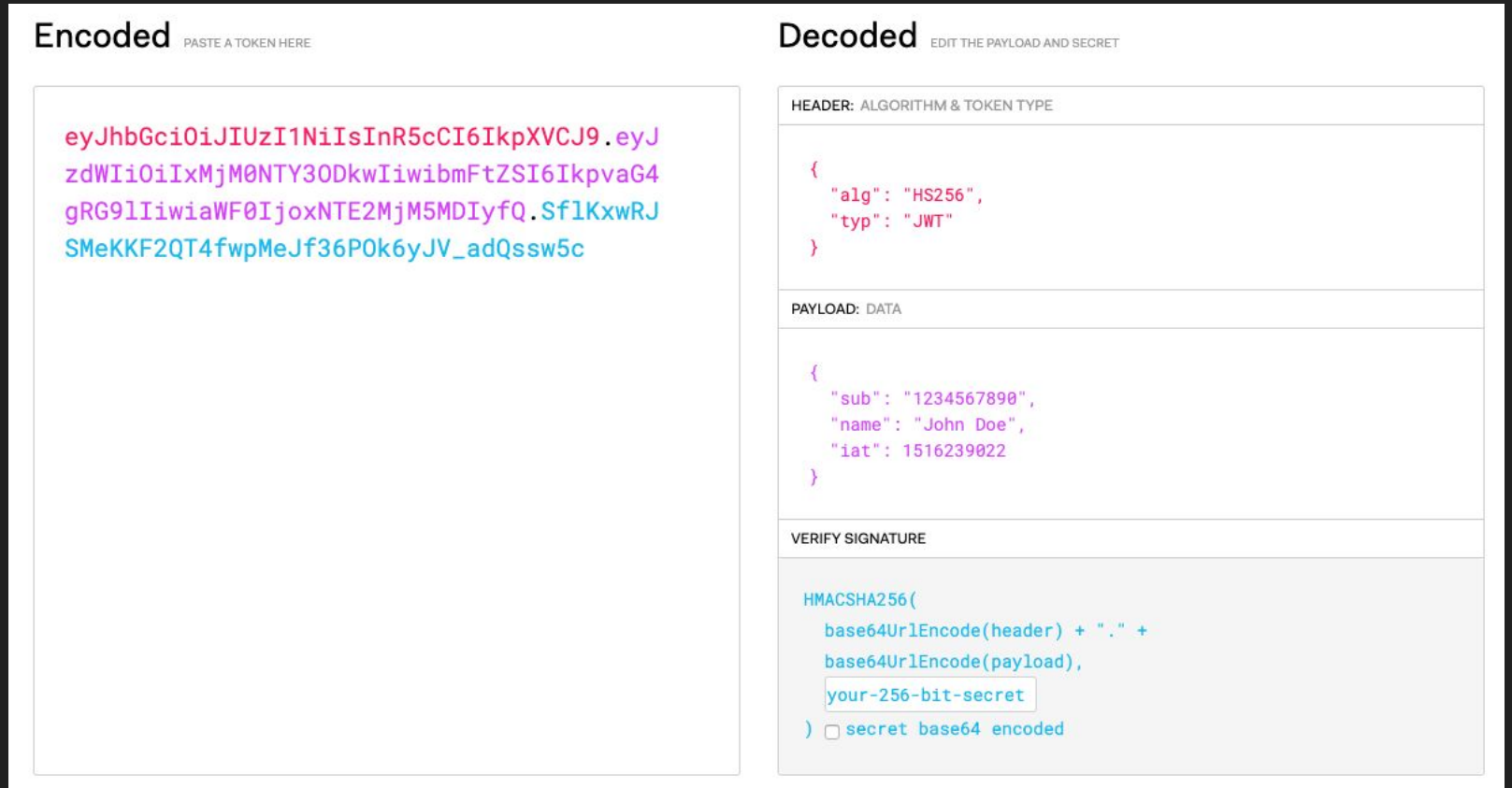
JWT Token Structure

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Ik1rVXlOVEF4TXpFd1EwUkNSVGxHT0
VSQk9EbEVRekI4UTBJeVFUazFNak0yTURSR1JUWXhNdyJ9.eyJodHRwczovL3ZzbS50Y
XNrdG9wLm5ldC9pc1N5c3RlbUFkbWluIjpmYWxzZSwiaHR0cHM6Ly92c20udGFza3RvcC5
uZXQvaXNFbWFpbFZlcmlmaWVkIjp0cnVlLCJpc3MiOiJodHRwczovL2xvZ2luLnZzbS50YXN
rdG9wLm5ldC8iLCJzdWIiOiJhdXRoMHw1ZDQzNGI5YTEwMzVhODBjYWRkYjk1MTMiLCJh
dWQiOlsiaHR0cHM6Ly92c20udGFza3RvcC5jbG91ZC9hcGkvaW5zdGFuY2VzL3F1ZXJ5Iiw
iaHR0cHM6Ly90YXNrdG9wLmF1dGgwLmNvbS91c2VyaW5mbyJdLCJpYXQiOjE1NjY5NDE
yMTEsImV4cCI6MTU2Njk0ODQxMSwiYXpwIjoicXJsbUJ3WUhyOTVUYVpQNG1UTW9WY3
JqYmpMMUJ3VlMiLCJzY29wZSI6Im9wZW5pZCBwcm9maWxlIGVtYWlsIHJlYWQ6YWxsIn
0.tD5cctFNQAHX9_jSGYz3ujWrqubAqPrRJ8bCJ_A4ouTA3W6rFNR_wo_JqW7PzoTlDbZ_1
hLIav94UEmW3pgwZmRbV01VlzxCDQC_25ZNlf1DLYQXhQV3P-e3qJbFtU3rGqyc8TDGSy
CgllJtfufQE8It4UDMHoSqAkEp1aFlxYlwxexduWygoa9S3veDws7kzO-SAntK1-eBTXFAe3Q
p5KRvcV5nJYQBbI1nTaKD-eRu9ZqIQ6JtgIzvPLQZQJ_u4kRi4FN_7BrDjEYLrNixZZjvlYwSg
2bfDV8yFXdeygwt9LEHfdYVOcQNYTNyMLpf-fkaXIAKrQWSs76k39L_kg

JWT Token Structure

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Ik1rVXlOVEF4TXpFd1EwUkNSVGxHT0VSQk9EbEVRekI4UTBJeVFUazFNak0yTURSR1JUWXhNdyJ9.eyJodHRwczovL3ZzbS50YXNrdG9wLm5ldC9pc1N5c3RlbUFkbWluIjpmYWxzZSwiaHR0cHM6Ly92c20udGFza3RvcC5uZXQvaXNFbWFpbFZlcmlmaWVkIjp0cnVlLCJpc3MiOiJodHRwczovL2xvZ2luLnZzbS50YXNrdG9wLm5ldC8iLCJzdWIiOiJhdXRoMHw1ZDQzNGI5YTEwMzVhODBjYWRkYjk1MTMiLCJhdWQiOlsiaHR0cHM6Ly92c20udGFza3RvcC5jbG91ZC9hcGkvaW5zdGFuY2VzL3F1ZXJ5Iiwia HR0cHM6Ly90YXNrdG9wLmF1dGgwLmNvbS91c2VyaW5mbyJdLCJpYXQiOjE1NjY5NDEyMTEsImV4cCI6MTU2Njk0ODQxMSwiYXpwIjoicXJsbUJ3WUhyOTVUYVpQNG1UTW9WY3JqYmpMMUJ3VlMiLCJzY29wZSI6Im9wZW5pZCBwcm9maWxlIGVtYWlsIHJlYWQ6YWxsIn0.tD5cctFNQAHX9_jSGYz3ujWrqubAqPrRJ8bCJ_A4ouTA3W6rFNR_wo_JqW7PzoTlDbZ_1hLIav94UEmW3pgwZmRbV01VlzxCDQC_25ZNlf1DLYQXhQV3P-e3qJbFtU3rGqyc8TDGSyCgllJtfufQE8It4UDMHoSqAkEp1aFlxYlwxexduWygoa9S3veDws7kzO-SAntK1-eBTXFAe3Qp5KRvcV5nJYQBbI1nTaKD-eRu9ZqIQ6JtgIzvPLQZQJ_u4kRi4FN_7BrDjEYLrNixZZjvIYwSg2bfDV8yFXdeygwt9LEHfdYVOcQNYTNyMLpf-fkaXIAKrQWSs76k39L_kg

JWT.IO


Encoded
PASTE A TOKEN HERE
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
Decoded
EDIT THE PAYLOAD AND SECRET
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
) secret base64 encoded

JWT Token Header


```
{
  "typ": "JWT",
  "alg": "RS256",
  "kid": "MkUyNTAxMzEwQ0RCRTlGOERBODlEQzIxQ0IyQTk1MjM2MDRGRTYxMw"
}
```

JWT Token Payload (Claims)


```
{
  "https://vsm.tasktop.net/isSystemAdmin": false,
  "https://vsm.tasktop.net/isEmailVerified": true,
  "iss": "https://login.vsm.tasktop.net/",
  "sub": "auth0|5d434b9a1035a80caddb9513",
  "aud": [
    "https://vsm.tasktop.cloud/api/instances/query",
    "https://tasktop.auth0.com/userinfo"
  ],
  "iat": 1566941211,
  "exp": 1566948411,
  "azp": "qrlmBwYHr95TaZP4mTMoVcrjbjL1BwVS",
  "scope": "openid profile email read:all"
}
```

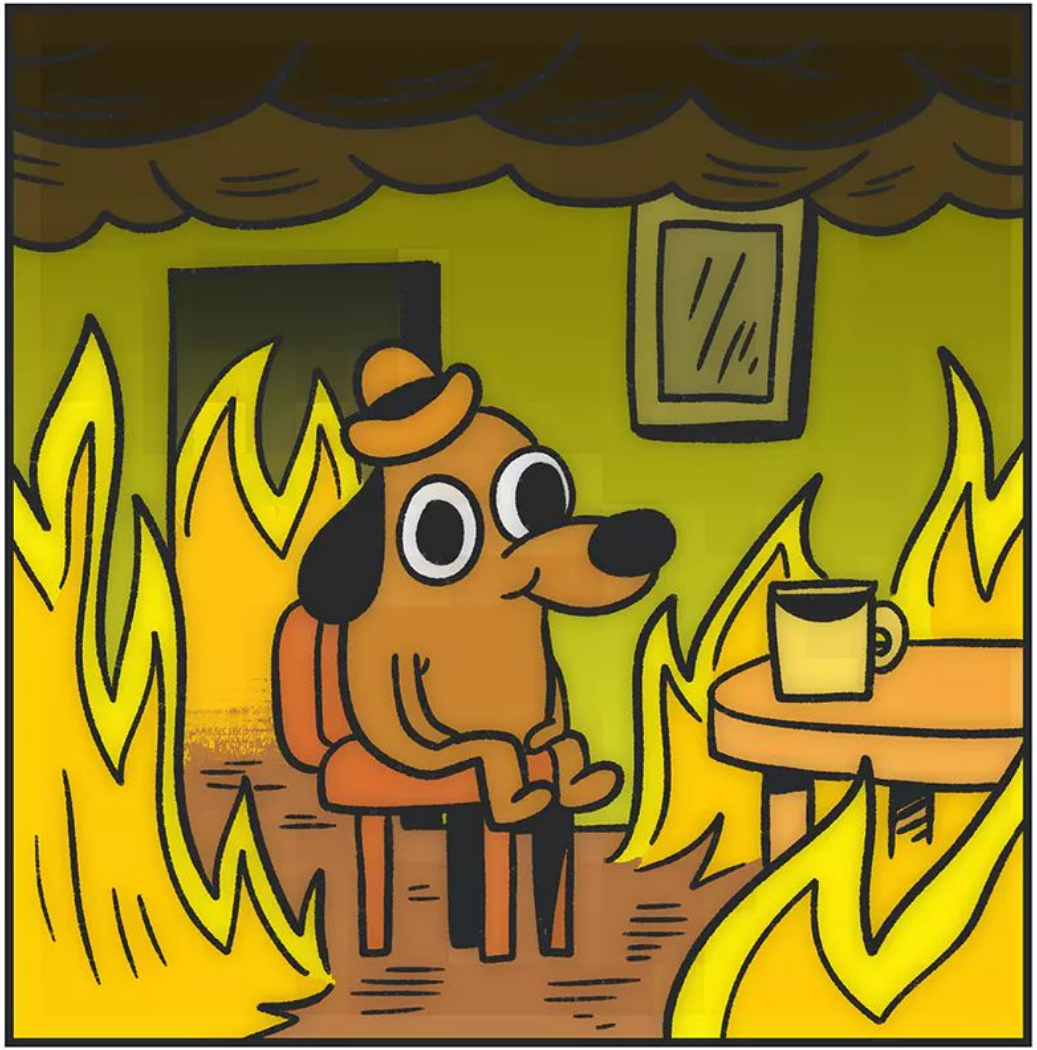
JWT Token Signatures

"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

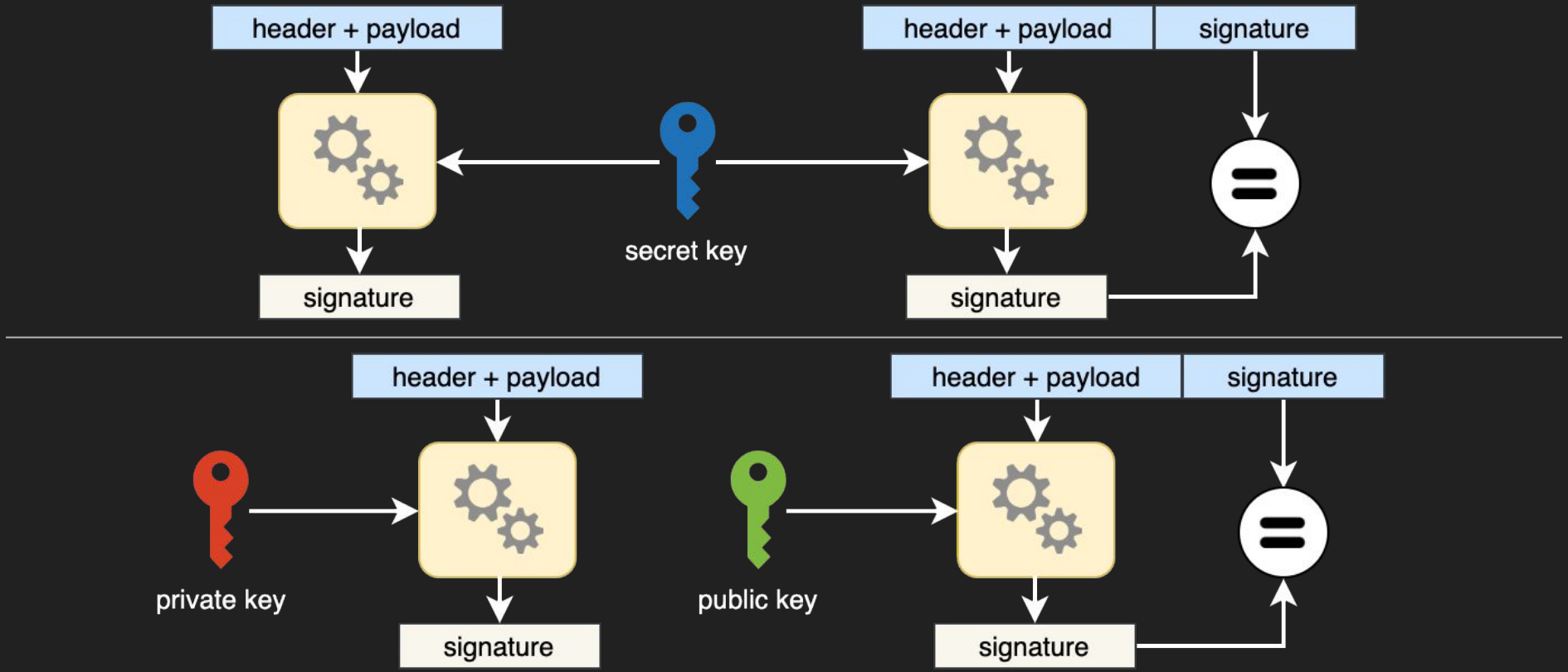
JWT Security

- Token content is plaintext!
  - Separate JWE standard for encryption - RFC 7516
- JWT token cannot be invalidated by itself
  - logout
  - compromised accounts
  - password changes
  - permission changes
  - user de-provisioning
- Stateless backends require careful consideration of token lifetime
- JWT header has to be validated, in particular only allowing specific algorithms

DEMO



Signature Algorithm Confusion...



JWT Secret Brute Forcing

RFC 7518 (JSON Web Algorithms) states that "A key of the same size as the hash output (for instance, 256 bits for "HS256") or larger MUST be used with this algorithm."

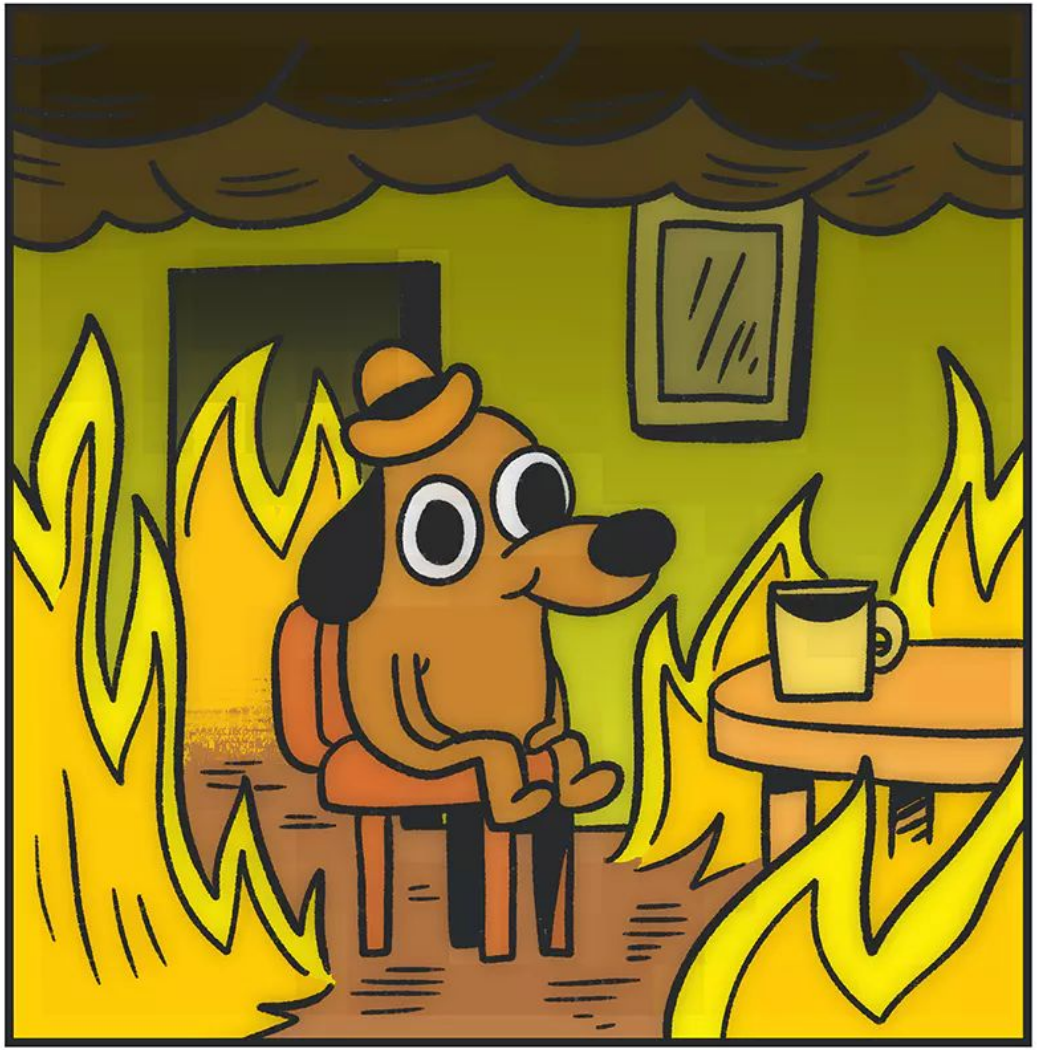
Shorter keys can be brute forced.

Tools:

- https://github.com/brendan-rius/c-jwt-cracker
- Hashcat support, hash id 16500
- John the Ripper support

DEMO

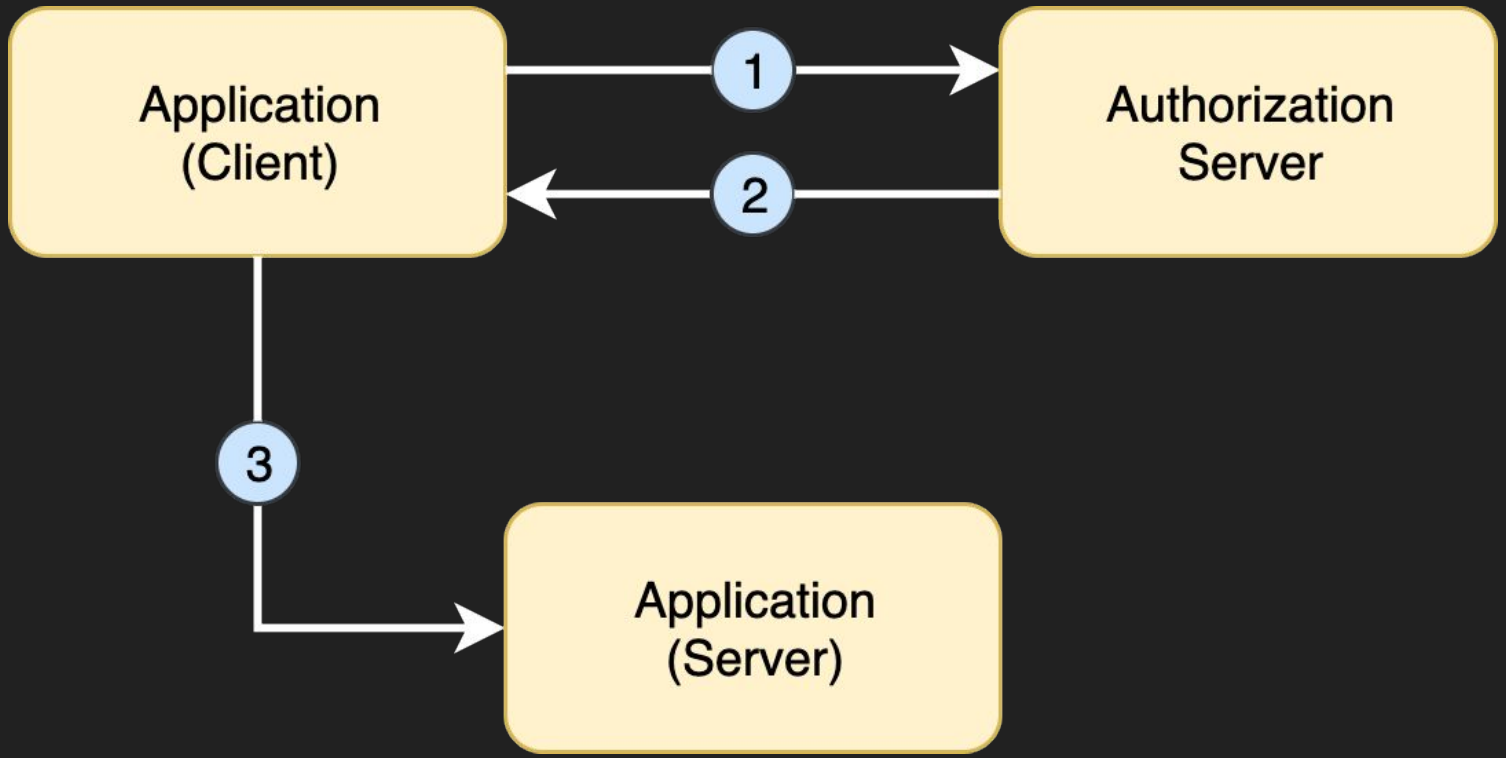
based on:
https://github.com/gluckzhang/ctf-jwt-token



JWT Security

- JWT storage - cookie XSS protections (HttpOnly & secure flags) are not available for browser local/session storage.
- Best practice - memory-only JWT token handling.
- Protection of the crypto keys (server side).
- Protection against CSRF - it’s not JWT tokens, it’s about how you use them.

Memory-only JWT & usability



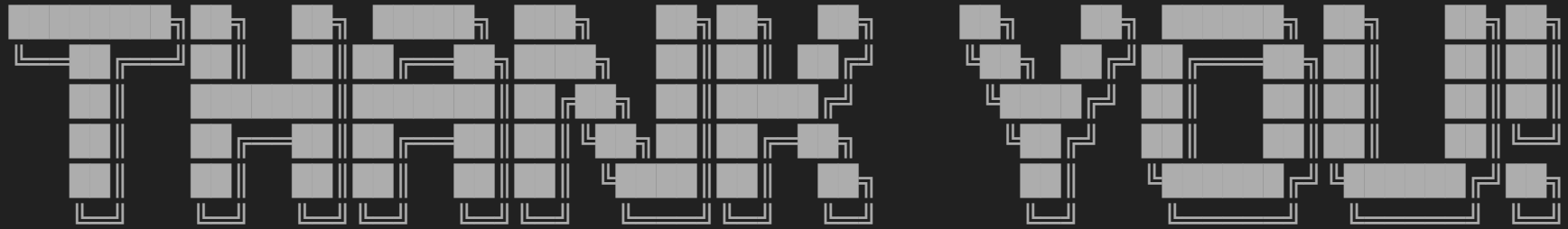
JWT Token Verification

- Header
  - **alg** - only allow specific algorithm(s)
  - ***kid*** *- check if present*
- Verify signature
- Validate payload
  - ***iat*** *- issued before current time*
  - **exp** - hasn't expired
  - **iss** - valid issuer
  - **aud** - valid "audience"
  - ***azp*** *- valid client ID if present*
- Validate custom "claims"

JWT Security

Most secure (though not always practical) use of JWT tokens:

- tokens used for authorization, but not session management
- short lived (few minutes)
- expected to be used once (confirm authentication/authorization and get a session ID)


THANK YOU