JWT

JSON Web Token

Intro

Estándar abierto para el pasaje de “claims” (información de seguridad) entre 2 partes

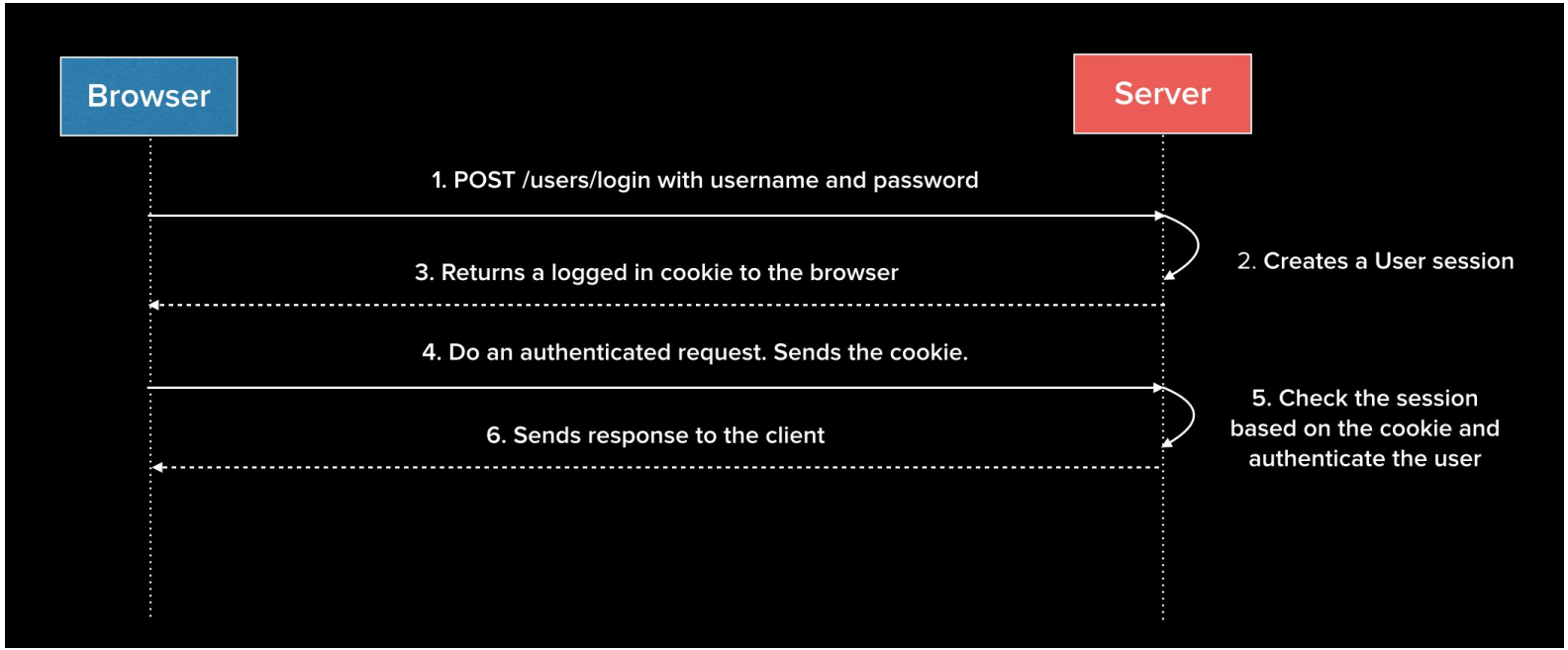
- URL-safe: Se puede usar/pasar en query strings, forms, cookies o HTTP Headers.
- “Claims” codificados como un objeto JSON que es firmado digitalmente (usando el estándar JWS)
- Autocontenido: Toda la información necesaria se encuentra dentro del mismo token.
- Parte de JOSE: “Javascript Object Signign and Encryption”

Intro

Mayormente usado en aplicaciones web

- Single sign-on
- Stateless apps
- Action links
- Webhooks
- Token-based Auth

Introduccion - Auth con session y cookie

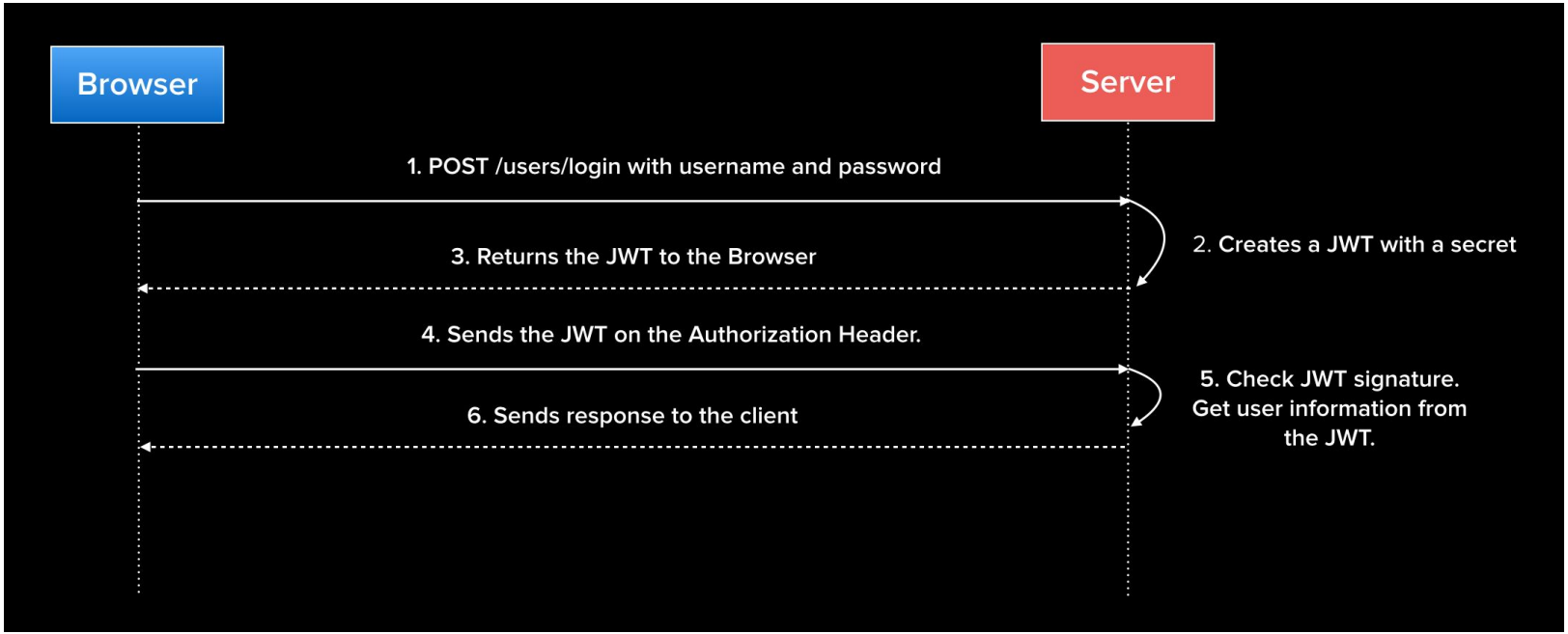


Intro - Auth con session y cookie

Algunos problemas (modernos)

- Tenemos que guardar la session en algun lugar del server
- En algunos casos de mobile el manejo de cookies es complicado
- CSRF
- CORS

Intro - Auth con JWT



Intro - Auth con JWT

Algunas ventajas

- AJAX a cualquier server (no CORS)
- Stateless: Escalabilidad de los servers
- Esquema de autenticación desacoplado
- Simplifica lo mobile
- No hace falta pensar en CSRF
- Performance? Discutible si calcular hash mas rapido que búsqueda a la DB

Estructura - JWT

---

3 secciones

- header, payload y signature
- Codificadas en base64

aaaaaa.bbbbbb.cccccc

HEADER - PAYLOAD - SIGNATURE

Estructura - Header

Usualmente formada por 2 partes, en formato JSON

- “typ”: “JWT”
- “alg”: identificador del algoritmo de hash usado

```
{
  “typ”: “JWT”,
  “alg”: “HS256”
}
```

Estructura - Payload

Contiene

- La información a transmitir
- La información relacionada con el token mismo
- En formato JSON

```
{
  “iss”: “Coquelux Web App”,
  “name”: “nahuel”,
  “admin”: false
}
```

Estructura - Payload

Existen “claims” que son parte del estándar:

- “iss”: issuer
- “sub”: subject
- “aud”: audience
- “exp”: expiration
- “nbf”: not before
- “iat”: issued at
- “jti”: JWT Id

Estructura - Signature

Utilizado para verificar la autenticidad del JWT. Existen difrentes algoritmos, algunos utilizan a **shared secret** (HMAC) y otro **public-private key** (RSA).

```
signature = HMACSHA256(
  base64UrlEncode(header) + “.” + base64UrlEncode(payload),
  SECRET
)
```

Estructura - JWT

JWT = header + “.” + payload + “.” + signature

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4i
OnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

header:
```
{
  “alg”: “HS256”,
  “typ”: “JWT”
}
```

payload:
```
{
  “sub”: “1234567890”,
  “name”: “John Doe”,
  “admin”: true
}
```

signature:
```
HMACSHA256(
  base64UrlEncode(header) + “.” +
  base64UrlEncode(payload),
  ‘secret’
)
```

JWT - Ejemplos en Python

---

1) encode.py
2) decode.py
3) validate.py

JWT - Ejemplos en Python

API Proxy server.py

1) login
2) get payload
3) get menus
4) get user info

JWT - Conclusiones

- Es un estándar
- Es simple
- Librerías para cualquier lenguaje
- Moderno
- Se pronuncia jot :P

Preguntas? :)

JWT - Apendix

- http://jwt.io/
- https://tools.ietf.org/html/rfc7519
- http://angular-tips.com/blog/2014/05/json-web-tokens-introduction/
- https://scotch.io/tutorials/the-ins-and-outs-of-token-based-authentication
- https://scotch.io/tutorials/the-anatomy-of-a-json-web-token
- https://speakerdeck.com/jpadilla/djangocon-json-web-tokens