ToiletPaper #76

JSON Web Token

Andreas Scharf / Senior Software Architect / Business Division New Business

✕ Problem

You don't want to use session cookies and handle state with sticky sessions in load balancers.
You want to create a **secure, self-contained token** for authentication or information exchange.

✓ Solution

JSON Web Token (JWT) is an open standard (RFC 7519) to securely communicate a JSON object.
Typically, the user has to log in successfully and will receive a token for API calls in return.
The token contains three parts: Header, payload and signature. Because the payload contains all required information, no database query is necessary for authentication. This is awesome for scaling stateless backend architectures!

A JWT can be decoded and read by anybody. In fact, this is useful for the client and debugging. The payload is not encrypted (see JWE for encryption), but valid signatures can only be created knowing a secret. Every time a token is received, an integrity check has to validate the signature to make sure the token was not manipulated. Afterwards its content is seen as trustworthy.

Expiration control can be implemented by including time related claims like *iat* ("Issued At"), *nbf* ("Not Before") and *exp* ("Expiration Time"). An expired JWT is still valid, but because it's not possible to manipulate entries without breaking the signature, the server side integrity check will read the tokens expiration date and deny access.

A disadvantage is that it is complicated to revoke a token. A common solution is blacklisting. Short validity durations help.

➔ Example

In this example we're creating a token issued by "jambit" for the subject "jambitee". The expiration time is set to limit validity of the token. We added the user's role, too. Any information can be part of the token, but it should be as minimal as possible to keep the token compact. The output is three Base64 encoded strings separated by dots. In practice you will use a library - available for many programming languages - to create a token or read its claims.

Header	Payload	Signature
The header contains the token **type** and the signature **algorithm**.	Public claims like *iss* ("Issuer"), *sub* ("Subject") and *exp* ("Expiration Time") are defined in the standard. Additional private claims can be used to fit your use case (e.g. **roles**).	To verify consistency Header and Payload content are signed with **HMAC** or public/private key pair using **RSA**.

<pre>{
  "alg": "HS256",
  "typ": "JWT"
}</pre>
	<pre>{
	  "iss": "jambit",
	  "sub": "jambitee",
	  "exp": 1412345678,
	  "roles": ["ROLE_USER"]
	}</pre>
		HMACSHA256(
		  base64UrlEncode(header) + "." +
		base64UrlEncode(payload),
		secret
		)

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJqYW1iaXQiLCJzdWIiOiJqYW1iaXRlZSIsImV4cCI6MTQyMjc3OTYzOCwicm9sZXMiOlsiUk9MRV9VU0VSIl19.PXXM0IdEJlWM1q0xDLXGyhImsUtvpwfQrQ1uPy2R2m4

✚ Further Aspects

- JWT.io – Project website with online debugger and libraries
- Identity APIs like Auth0 and Stormpath are supporting JWT
- Available Libraries: .NET, C, Clojure, Crystal, D, Delphi, Elixir, Go, Haskell, Java, JavaScript, Lua, Node.js, Objective-C, Perl, PHP, Python, Q, Ruby, Rust, Scala, Swift More information:

https://tools.ietf.org/html/rfc7519 https://auth0.com/learn/json-web-tokens/
https://medium.com/@berto168/jwt-the-basics-and-more-63e7f1fc43c6
https://developer.atlassian.com/static/connect/docs/latest/concepts/authentication.html