Introducció a l'autenticació amb tokens


A l'hora de protegir unes certes rutes o seccions d'una aplicació web, podem utilitzar diversos mecanismes. En el que a les aplicacions web "tradicionals" es refereix (és a dir, aquelles que serveixen contingut visible en un navegador, com per exemple, i sobretot, contingut HTML), el mecanisme d'autenticació per antonomàsia és l'autenticació basada en sessions.

No obstant això, quan volem estendre l'aplicació web més enllà de l'ús d'un navegador, i permetre que altres tipus de clients es connecten al *backend* (per exemple, aplicacions d'escriptori, o aplicacions mòbils), l'autenticació basada en sessions es queda curta o no serveix, i és necessari recórrer a altres mecanismes més universals, com l'autenticació amb tokens. És el tipus d'autenticació més utilitzat en serveis REST.

1. Fonaments de l'autenticació amb tokens

Un **token** per si mateix és una cadena de text que manca de significat. Però, combinat amb unes certes claus, és un mecanisme per a protegir les nostres aplicacions d'accessos no permesos. L'autenticació basada en tokens és un mètode mitjançant el qual ens assegurem que cada petició a un servidor ve acompanyada per un token signat, que conté les dades necessàries per a verificar que el client ja ha sigut validat prèviament.

El mecanisme emprat per a l'autenticació amb tokens és el següent:

1. El client envia les seues dades d'autenticació al servidor (típicament un login i password).
2. El servidor valguda aqueixes credencials contra alguna mena d'emmagatzematge (normalment una base de dades). En el cas que siguen correctes, genera un token, una cadena codificada que permet identificar a l'usuari. Aquest token s'envia al client, normalment com a dades de la resposta. Internament, pot contindre alguna dada que permeta identificar a l'usuari, com el seu login o e-mail.

> **NOTA:** No convé afegir en el token (ni tampoc en les sessions) informació molt confidencial, com el password, per exemple, ja que pot ser fàcilment descodificable. Això no vol dir que el token siga un mecanisme insegur d'autenticació, ja que el servidor utilitza una paraula secreta per a xifrar una part del token, i així poder verificar que és correcte, però la resta del token sí que queda més descobert.

3. El client rep el token i l'emmagatzema d'alguna forma localment (mitjançant mecanismes com `localStorage` en JavaScript o similars en altres llenguatges). Davant cada nova petició que es faça, el client reenvia dit token en les capçaleres de la petició, perquè el servidor verifique que és un client autoritzat.

Normalment als tokens se'ls assigna un temps de vida o una data de caducitat (que poden ser minuts, dies, setmanes... depenent del que ens interesse i del tipus d'aplicació). En cada nova reconnexió, el temps de vida es pot renovar (no és una cosa automàtica, haurem de fer-ho nosaltres), i si passa més d'aqueix temps estipulat sense que el client intente connectar, se li sol·licitarà de nou que s'autentique.

Se sol emprar l'estàndard JWT (JSON Web Token), que defineix una forma compacta de transmetre aquesta informació entre client i servidor emprant objectes JSON.

2. Estructura d'un token JWT

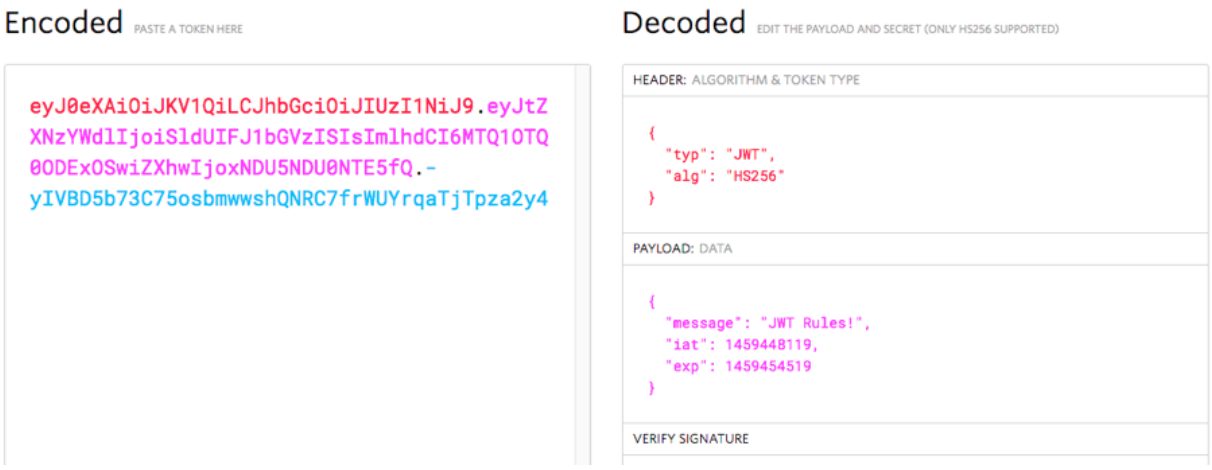
Un token JWT consisteix en tres parts:

- **Capçalera (*header*)**: amb informació sobre el tipus de token generat (en aquest cas, JWT), algorisme d'encriptació (HS256, per exemple), etc.
- **Càrrega (*payload*)**: conté la informació codificada pel token (login, rols, permisos concrets, caducitat, etc), la qual cosa es coneix com a declaracions (*claims*). Convé no enviar informació confidencial en aquest element, ja que, com veurem, és fàcilment descodificable.
- **Signatura (*signature*)**: s'empra per a validar el token i protegir-ho enfront de manipulacions. Aquesta signatura es genera mesclant tres elements: la capçalera, el payload i una paraula secreta.

Es generarà una cadena en format *base64* amb tota la informació del token. Per exemple:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJtZXNzYWdlIjoiSldUIFJ1bGVzISIsImlhdCI6MTQ1
OTQ0ODExOSwiZXhwIjoxNDU5NDU0NTE5fQ.-yIVBD5b73C75osbmwwshQNRC7frWUYrqaTjTpza2y4
```

Si peguem aquesta cadena en algun processador de tokens, com el de jwt.io, podrem veure descodificades la capçalera i el payload:



Això vol dir que no existeix una encriptació per a les dades que s'envien (capçalera i payload), es poden descodificar fàcilment des de *base64*. Però sense conéixer la paraula secreta que genera la signatura, no es pot validar que aqueix token siga correcte. Aqueixa paraula només la coneix el servidor. En el cas de l'exemple anterior, si utilitzem la paraula "L3@RNJWT" podrem obtindre la validació del token en la pàgina anterior.

És a dir, la tercera part del token s'empra pel servidor per a, una vegada obtingudes la capçalera i el payload (descodificant-les des de *base64*) i unint a elles la paraula secreta, comprovar que la cadena generada coincideix amb aqueixa tercera part. Si és així, el token és autèntic.

3. Avantatges i inconvenients de l'autenticació amb tokens

El mecanisme d'autenticació amb tokens ofereix alguns avantatges respecte al tradicional mètode per sessions:

- Els tokens no tenen estat (*stateless*), cosa que significa que el servidor no ha d'emmagatzemar enlloc el registre d'usuaris autenticats per a comprovar si qui entra ha sigut autoritzat abans. La pròpia cadena que s'envien client i servidor conté tota la informació. Per contra, la informació de les sessions (o els identificadors de cada sessió) sí que s'emmagatzema en el servidor.
- Qualsevol aplicació client suporta tokens, la qual cosa suposa que podem emprar aquest mateix mecanisme per a autenticar una aplicació d'escriptori, mòbil o web contra el mateix *backend*.
- Els tokens permeten especificar informació addicional d'accés, com a rols d'usuari, permisos concrets a recursos, etc.

No obstant això, per a poder implementar una autenticació basada en tokens en una aplicació tradicional de navegador, necessitem que el client no siga un client "ximple", que es limite a renderitzar el contingut HTML i poc més. És necessari cert pre-processament JavaScript en la part client per a enviar el token al servidor en cada petició, així com per a recollir el token generat pel servidor quan ens autentiquem. Això fa imprescindible l'ús d'unes certes llibreries en el client, com jQuery, Angular, React o Vue, entre altres.