CONSIDERAZIONI SULL'USO DEL JSON WEB TOKEN (JWT)

Autenticazione senza JWT

Per effettuare un'operazione di login, una delle soluzioni più comuni in passato, ma in realtà molto utilizzata ancora oggi, consiste nel creare una sessione tra il server e il client, ovvero un canale di comunicazione attraverso il quale vengono inviate le credenziali dell'utente e che resta in vita, con all'interno tutte le informazioni necessarie, compreso l'identificativo dell'utente, per tutta la durata del collegamento.

Senza entrare nel merito di considerazioni di sicurezza, se si rende necessario bilanciare il carico di accessi su più server, il meccanismo di gestione delle credenziali attraverso la sessione diventa macchinoso.

Esistono anche altre possibili soluzioni per effettuare l'attività di login senza ricorrere alla memorizzazione temporanea dei dati nella sessione, ma generalmente presuppongono la memorizzazione di informazioni in un database.

In questo modo si può bilanciare il carico su più server, ma si intensifica il colloquio tra i vari server e il DB, oltre a diminuire il livello di sicurezza perché aumentano le possibilità che il database sia oggetto di attacchi.

Nel tempo sono stati sviluppati per l'autenticazione metodi di dialogo tra server e client più efficienti e sicuri, come il SAML o il JWT, ma quest'ultimo ormai si è affermato come standard tra i più diffusi.

Cos'è il JWT

Json Web Token (comunemente indicato con l'acronimo JWT) è uno standard open (RFC 7519) nato nel 2015 per implementare un dialogo tra client e server che permetta ai due interlocutori di "riconoscersi" e scambiarsi informazioni accessorie in maniera certa.

Il server che si occupa dell'autenticazione scrive in un oggetto, opportunamente costruito, chi è l'utente e altre informazioni utili; queste informazioni vengono incapsulate in un gettone (token) che viene restituito al client.

Il client che riceve il token dovrà verificarne la correttezza, ma non potrà modificarlo perché ne invaliderebbe l'integrità e il server lo rifiuterebbe al primo tentativo di utilizzo.

Per le successive chiamate al server, il client fornirà il token, permettendo al server di identificare l'utente che sta effettuando la chiamata.

Il server, quando lo riceve, si assicura che il token sia stato firmato e autentificato in maniera corretta. Essendo contenute nel token tutte le informazioni necessarie per l'identificazione dell'utente richiedente, il server avrà le informazioni di autenticazione direttamente nel token stesso e non necessiterà di sessioni o accessi al DB. Tale caratteristica rende anche possibile l'utilizzo del medesimo gettone su server differenti, necessario nel caso di bilanciamento del carico di lavoro tra più server.

Se le informazioni nel token sono scritte in maniera adeguata e se il token viene verificato ad ogni utilizzo, risulta evidente che il metodo JWT rappresenta una pregevole sintesi tra semplicità d'uso e sicurezza.

Com'è strutturato il JWT

Un JWT è una struttura divisa in tre parti, ognuna delle quali costituita dalla codifica in Base64 di varie informazioni:

- un **header**, struttura JSON contenente dati descrittivi;
- un **payload**, struttura JSON contenente i dati di autenticazione;
- una **signature**, contenente un'impronta che certifica quanto passato nelle due parti precedenti.

L'**header** contiene informazioni sulla tipologia del token (che nello standard JWT è necessariamente JWT) e sul tipo di algoritmo di cifratura utilizzato per la signature.

Nel **payload** sono presenti informazioni relative all'entità (tipicamente un identificativo dell'utente) che ha richiesto il token e altri dati di interscambio. Si evidenzia che questi dati non sono criptati, quindi non devono essere informazioni riservate.

La **signature** è la codifica in base 64 dell'header e del payload uniti da un "."; alla stringa così ottenuta si applica successivamente l'algoritmo di cifratura indicato nell'header utilizzando una chiave segreta.

La chiave segreta potrebbe essere teoricamente nota al client e al server, ma la sicurezza è nettamente maggiore se si utilizzano coppie di chiavi asimmetriche pubbliche/private.

Esemplificazione di un token JWT

HEADER

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

Nell’header sono racchiuse le informazioni principali relative all’algoritmo utilizzato per preparare la firma criptata e sulla tipologia di token.

L’oggetto json viene codificato usando la funzione encodebase64 e la parte di header del token risultante è la seguente:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
```

PAYLOAD

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

Le informazioni inserite in questo payload di esempio sono che l’utente loggato si chiama John Doe, il suo ruolo è amministratore e il suo id è 1234567890.

Il payload viene anch'esso codificato con la funzione encodebase64, ottenendo la seconda parte del token che è la seguente:

eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9

Si ricorda che le informazioni sono "in chiaro", perché possono essere decodificate con la funzione decodebase64.

SIGNATURE

HMACSHA256(
Base64UrlEncode(header) + "." + Base64UrlEncode(payload), MY-KEY-SECRET
)

La signature è la firma con cui il server certifica che questo messaggio è stato scritto da lui e che non è stato modificato da nessun altro.

La firma non è altro che il risultato di una funzione hash 256 che prende in input la codifica base64 dell'header concatenandola con un punto alla codifica base64 del payload, il tutto codificato con la "chiave segreta" che solo il server conoscerà!

Il risultato finale è la concatenazione di queste 3 parti appena viste:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA950rM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Il server quindi restituirà questo token al client.

Il client dovrà comunicare al server il token ricevuto per tutte le successive chiamate in cui è richiesta l'autenticazione.

Il server riceverà il token ed estrapolerà il payload (in cui sono contenute le informazioni sull'utente loggato), ma prima verificherà la validità del token.

Aumento della sicurezza nell'uso del JWT

Il meccanismo di funzionamento del JWT, come si è visto sopra, è un meccanismo che garantisce un ottimo livello di sicurezza. L'utilizzo di alcuni accorgimenti può renderlo ancora più efficace.

Un primo accorgimento è l'utilizzo di un canale sicuro per la comunicazione tra client e server. Per le applicazioni web e le app l'utilizzo del protocollo https è un prerequisito irrinunciabile.

Un secondo accorgimento è l'utilizzo, al posto di una chiave di sicurezza unica nota a server e client, di una chiave asimmetrica, cioè formata da una chiave pubblica e una chiave privata.

In Autentica Admin (l'applicazione di amministrazione di Autentica), la coppia di chiavi asimmetriche viene generata quando un amministratore di Autentica crea un nuovo progetto. La chiave privata viene utilizzata per firmare digitalmente il token e resta riservata, cioè nota solo al server. La chiave pubblica viene mostrata fra i dati del progetto e deve essere utilizzata dal client per verificare la firma digitale dei token prodotti.

Autentica Admin, però, permette anche all'amministratore di utilizzare per ogni progetto un proprio certificato (contenente chiave privata e pubblica); se si utilizza questa opzione, il certificato deve essere in corso di validità, dato che la sua verrà verificata ad ogni utilizzo.

Un terzo accorgimento è quello che il client effettui sempre la verifica della firma, prima di utilizzare i dati contenuti nel token, che d'altra parte non possono essere modificati se non in possesso della corretta chiave privata, pena la compromissione della signature.

Un quarto accorgimento messo in pratica da Autentica è inserire nel payload del token un parametro contenente un identificativo univoco (parametro jti); in questo modo non possono esistere due token uguali.

Un altro utile accorgimento di Autentica è includere un timestamp di scadenza del token (parametro exp); l'utilizzatore del token, oltre a verificare la signature, dovrebbe rifiutare token scaduti. È utile precisare che il periodo di validità di un gettone è un parametro modificabile a livello di Autentica Admin dall'amministratore.

Un ulteriore accorgimento è l'utilizzo di un parametro "nonce", che si consiglia generato random e sempre diverso, per permettere di creare sistemi di controllo per verificare di essere i reali "proprietari" del token.

Verificare se nel token ricevuto è contenuto il "nonce" fornito permette di sventare attacchi di tipo "replay-attack" e "a dizionario".

L'obiettivo di Autentica, quindi, è utilizzare lo strumento JWT, per sua natura standard, potente e flessibile, aggiungendo alcuni meccanismi di ulteriore controllo per incrementare il livello di sicurezza.

JWT e G.D.P.R.

È utile sottolineare anche i vantaggi che l'adozione del JWT porta a livello di compatibilità con le norme del GDPR.

Il fatto che le credenziali di autenticazione non risiedano più nello stesso db delle anagrafiche e, anzi, che siano addirittura su un altro server o, meglio ancora, su un server in un'altra infrastruttura, rende la possibilità di accesso indesiderato e di violazione dei dati un'eventualità praticamente impossibile.

Autentica conserva le credenziali su un server in maniera sicura, in un'infrastruttura in cloud (ospitata in una web farm su territorio italiano).

In un'architettura distribuita, l'utilizzo della tecnologia JWT, con gli accorgimenti adottati da Autentica che ne massimizzano la sicurezza, aumentano il livello di sicurezza per l'autenticazione e rendono sicuro il dialogo tra qualunque server e i suoi client.

In uno scenario pratico, infatti, il dialogo tra un client web o una app, ma anche un applicativo client server, e il server, sfruttano il meccanismo di autenticazione fornito da una terza parte (Autentica), che stacca un gettone con caratteristiche adeguate alle necessità di quella specifica applicazione, permettendo, oltre al riconoscimento sicuro in fase di autenticazione, anche il successivo dialogo sicuro.

Generazione Informatica Srl
tel: 055 6810299 - fax: 055 6585805
email: info@autenticazionesicura.it