JSON Web Token (JWT) benefits over a database session token

Asked 6 years, 3 months ago Active 1 year, 7 months ago Viewed 17k times

53

27

With a database session token system I could have a user login with a username/password, the server could generate a token (a uuid for example) and store it in the database and return that token to the client. Every request from thereon would include the token and the server would look up whether the token is valid and what user it belongs to.

Using JWT there would be no need to save anything to the database with respect to session/tokens thanks to the combination of the secret key kept on the server and the signed token the client keeps and sends with every request.

This is good but besides saving a database check each request (which would be fast anyway since it's just checking a hash table) it's not clear to me what the advantages are of using JWT. Can you anyone familiar with this explain? **Let's ignore cookies, it's specifically a database custom token as described above and JWT that I am trying to compare and understand the benefits**.

access-token jwt express-jwt json-web-token

edited Nov 4 '16 at 14:46
Waiting for Dev...
11.3k 5 41 55

asked Oct 6 '14 at 12:43
Skill M2
1,126 1 12 17

2 Answers

Active | Oldest | Votes

34

The main difference is the session storage size and lookup work required from the server:

- On the server side, JWT stores a **single key** in memory (or in config file) - called *secret key*. That key has two purposes, it enables creating new encrypted tokens and it also functions like a master key that "opens all locks"- or in real life verifies all tokens. As a result the server responds much faster to auth requests, because it doesn't matter if you have two or two million users logged in - the same number of records (one, that server key) will be used to authenticate all client requests.
- Traditional authentication that stores user sessions in a database, creates a record in the db for every single user, which results in **multiple keys**. So if you have two million users logged in, the server will create two million records and with each client request the server needs to locate the relevant session record in the database*.

JWT leaves it up to the client side to store and handle the entire session/user object. It actually makes much more sense because every client handles their own data only, so it doesn't cause

heavy lifting for the client side either.

As for what you wrote in your last paragraph, it's not just db calls that we save here. JWT is actually much more **scalable** because of its independent and lightweight nature, it doesn't fail as auth requests pile up and it allows the server to handle auth accross devices and services without managing sessions on the server side.

Security wise though, db sessions arguably have the upper hand: they can be more secure *because* of that latency, and are also less vulnerable to session hijacking after user logout.

*The db stored sessions method can be optimized with effective caching and by storing only the session id (as opposed to the entire user object) in a fast key/value server such as Redis. That said, I would still choose JWT method over db for most cases.

edited Mar 6 '17 at 7:54

answered Jun 23 '15 at 18:12

EranG

698 6 19

The first two points were something I touched upon in the question but since the extra storage and retrieval were both cheap for me I didn't consider this a huge advantage but I can see that it may indeed well be for some use cases, what's really interesting is the point you make about scalability of auth requests, this is an advantage, and the possibility of session hijacking is a disadvantage. I did end up using jwt btw but still need to figure out the best way to deal with potential session hijacking other than requesting password re-entry for sensitive apis. – Skill M2 Jun 29 '15 at 15:57

2 Actually, the hijacking is something that could occur with sessions too, it's just that with sessions you could clear out an individual users sessions where as with jwt you cannot, changing the secret key would log everyone out but cannot do it for an individual user. The scalability and lack of management associated with maintaining the sessions probably give jwt the edge. – Skill M2 Jun 29 '15 at 16:09

2 Have a look at this thread and answers dealing with jwt tokens security issues stackoverflow.com/questions/26739167/… the idea is to keep token lifecycle short and renew tokens often, or search google for "auth0 refresh token" (w/o quotes) for another solution – EranG Jun 29 '15 at 18:45

1 The first two points were all i needed to know. Very well explained, thanks! Before i didn't know how to handle the secret. – Tadej Jan 12 '17 at 12:08

I found this interesting cryto.net/~joepie91/blog/2016/06/19/… – Alex Mar 6 '17 at 12:54

2

A Json based token(JWT) overcomes the following problems:

1. Mobile issues: Native mobile apps seems to have problems working with cookies so if we need to query a remote API, maybe session auth is not the best solution.
2. CSRF issues: If you are following cookies way then you need to have CSRF to avoid cross site requests.

But JWT doesn’t use sessions, has no problems with mobile, it doesn’t need CSRF and it works very well with CORS too. If you dont have a valid token you can't do anything.

One more as this token gets stored at client local storage/session storage so you can pass these tokens to other clients as well but you have to share the same credential which you used to generate this JWT.

edited May 26 '19 at 6:11

answered Oct 6 '14 at 12:57

Priyanshu Shekhar
2,814 3 24 32

9 These are benefits JWT provides over using cookies, I understand that, but I am asking about session tokens stored in a db and generated by a server. – Skill M2 Oct 6 '14 at 13:07

1 Also, these are token (both JWT and opaque) benefits over cookies. – Waiting for Dev... Nov 7 '16 at 17:52