

How to use Google Identity to log in from multiple devices?

Asked 5 years, 8 months ago Modified 5 years, 8 months ago

Viewed 2k times



1



How can I use Google Identity platform as a login / registration system for my own service? Specifically - how can I do this and support login for same user from different devices?



Using for web service, nodejs without npm modules: passportjs / googleapis / google-auth-library.



My idea: User opens myClientApp/login page and clicks on GoogleLogin button which will ask him to authorize my service for specific scopes of his Google account. I then get the refresh token & access token and save it in DB, then send the refresh token to the client with cookie. Whenever I make a call to my own service API I send the refresh token from the cookie. As long as I have valid access token saved in my DB or the refresh token is not expired - I treat the user matching that refresh token as an active session for my service.

Security problems: cookies attacks, and the refresh token is easily accessed from the browser. Could use https / encryption and more methods to secure the cookie and

it's value. Still- someone could copy the cookie from one computer to another!

Multiple login problems: If the user login on different device, a new refresh token will be created. The previous device the user logged in to will now hold a wrong refresh token in the cookie...

Could the OpenID solve this? Do I need to use JWT?
What is the most secure way to use Google Identity login in my own service while supporting multiple devices login for the same user?

google-oauth

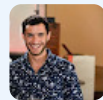
google-identity

Share

Improve this question

Follow

asked Mar 29, 2019 at 0:53



RanST

502 ● 2 ● 7 ● 23

1 Answer

Sorted by:

Highest score (default)



3



First, make sure that you really understand the security implications for what you want to do.

For example, NEVER send the Refresh Token to a client.

If you want to use the same tokens for the same client on multiple devices, you have a chicken and egg situation. How do you "authenticate" the user at each device. How



do you know that user "John" is actually user "John" but on a different device the first time?



Your goal is not to trade convenience for less security. Your goal should always be security first, no matter the inconvenience.

A better approach is to let Google authenticate and authorize a user on each device. They only have to do this once per device. Your backend systems keep track of the Refresh Token issued for each device. You can then generate the Access Tokens and Identity Tokens when needed - they expire after one hour anyways. Store a cookie on the user's device that identifies them to your system so that you can look up who they are, get the Refresh Token, create new Access Tokens, etc.

There is a limit to the number of Refresh Tokens that can be issued before the oldest ones are voided. I think the number is 50. This is usually not a problem. If a Refresh Token is invalid, just put the user back thru the authenticate process and store the new token.

Also provide the user with a sign-out method that removes all stored tokens in your system.

The cookie that you store on the client's devices should be opaque meaning that there is no stored information in the cookie and the cookie is only valid for that device and no other devices. This solves the stolen cookie moved to another device problem.

I will now touch on some of your questions:

My idea: User opens myClientApp/login page and clicks on GoogleLogin button which will ask him to authorize my service for specific scopes of his Google account.

Google OAuth does not work that way. You send the user to a URL, Google manages all display and input with the end user. Once everything is complete a callback URL on your server is called and you are passed a code. The exact details depend on the type of OAuth Flow that you are using.

I then get the refresh token & access token and save it in DB, then send the refresh token to the client with cookie.

During the OAuth Flow you will request the Access Token, Refresh Token and Identity Token. The Refresh Token is saved in the database. Never send this token to the client. Read my suggestion above about creating an opaque cookie that you send to the client.

Security problems: cookies attacks, and the refresh token is easily accessed from the browser. Could use https / encryption and more methods to secure the cookie and it's value. Still-

someone could copy the cookie from one computer to another!

Create an opaque cookie that is only valid for that device and no other devices. If a client sends you a cookie intended for a different device, consider this a problem and invalidate all cookies, tokens, etc for this user on all devices.

Multiple login problems: If the user login on different device, a new refresh token will be created. The previous device the user logged in to will now hold a wrong refresh token in the cookie...

I covered this issue above. Store the Refresh Token generated for each device in your DB. Consider each device / Refresh Token / cookie as a set.

Could the OpenID solve this? Do I need to use JWT? What is the most secure way to use Google Identity login in my own service while supporting multiple devices login for the same user?

By Open ID I think you mean Open ID Connect (OIDC). This is already integrated into Google OAuth and this is the part that generates the Identity Token.

Do I need to use JWT?

Google OAuth Tokens are generated from Signed JWTs. However for the most part you do not need to worry about the format of tokens. Google provides endpoints that validate and decode Google OAuth tokens.

What is the most secure way to use Google Identity login in my own service while supporting multiple devices login for the same user?

I covered this question in the first part of my answer above.

Share Improve this answer

Follow

answered Mar 29, 2019 at 1:51




[John Hanley](#)

81k ● 7 ● 112 ● 176

Thank you for the details response! About the login - I know that it works that way the user is sent to google URL and I managed to make it work for a single device. (I just really have hard time explaining myself correctly:P). --- I want to use the authorization code flow. I was probably ignoring the id_token. It is explained in many guides and other answers that this is the JWT - and as I understand JWT is the way to verify that "John" is actually himself form any device. So what I need to do is to use the id_token? – [RanST](#) Mar 29, 2019 at 11:09

The ID Token provides you identity information about the authenticated user. – [John Hanley](#) Mar 29, 2019 at 16:22

Just to be sure I understand correctly - I use of token to authenticate the user on each device? – [RanST](#) Mar 29, 2019 at 17:12

No, use an opaque cookie. I covered this in detail in my answer. Once the user authenticates on his device, store the Refresh Token and create an opaque cookie which you save in your server's DB. Store this cookie on the device. When the user comes back to your domain url, read the cookie and match to the user and device. – [John Hanley](#) Mar 29, 2019 at 17:41 

I'm sorry I'm bothering you again, but I don't quite understand what an opaque cookie is (read about it). What data should I save in it? Should the data be signed/encrypted? How do I match a specific device to a cookie? Is it similar to session cookies? – [RanST](#) Mar 29, 2019 at 18:03
