Different ways to Authenticate a Web Application

Vivek Madurai Feb 5, 2018 · 4 min read

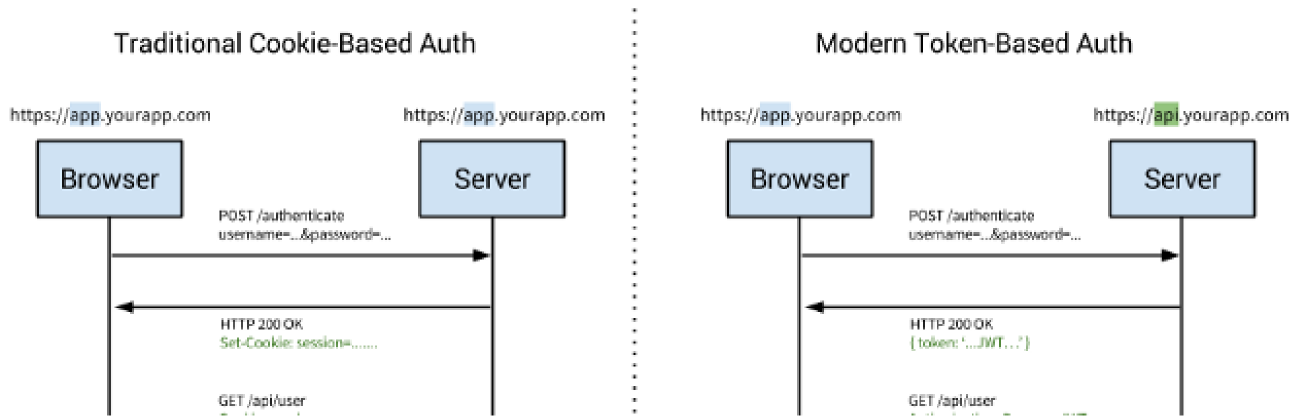
Authentication is common way to handle security for all applications. This is only way to answer the question "who you are?" to the application, when comes to stateless architecture or service oriented architecture we got lot of new concepts and technologies in the market. In this article we will learn how to handle authentication on RESTful APIs.

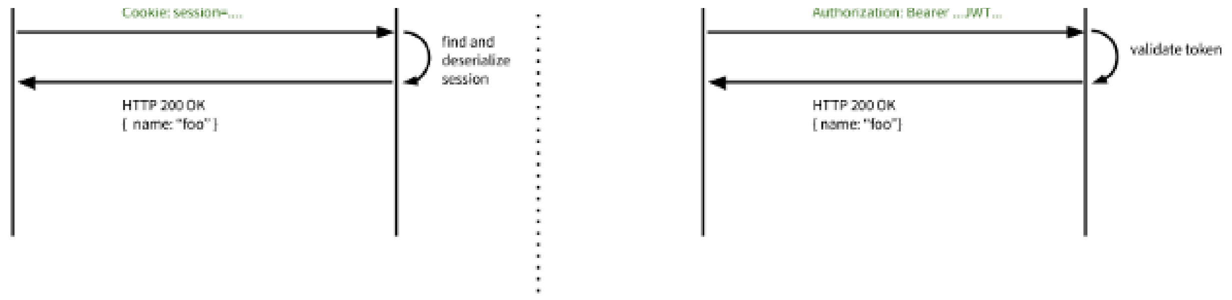
Authentication:

Lets start by listing the various ways through which we can achieve authentication,

- Cookie-Based authentication
- Token-Based authentication
- Third party access(OAuth, API-token)
- OpenId
- SAML

Cookie based authentication has been the default method for handling user authentication for a long time. From the below diagram you can clearly see the client posts the login credential to the server, server verifies the credential and creates session id which is stored in server(state-full) and returned to client via set-cookie. On subsequent request the session id from the cookie is verified in the server and the request get processed. Upon logout session id will be cleared from both client cookie and server.

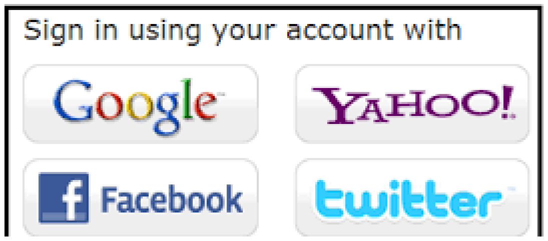


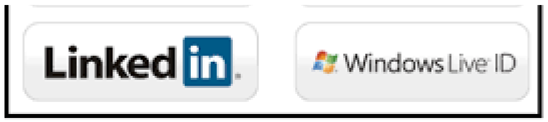


On the other hand **Token based authentication** is gaining in popularity because of the rise in single page applications(SPA) and statelessness(RESTful API's)of the application. There are different ways to implement token based authentication, we will focussing on most commonly used JSON Web Token(JWT). On receiving the credentials from client the server validates the credentials and generates a signed JWT which contains the user information. Note, the token will never get stored in server(stateless). On subsequent request the token will be passed to server and gets verified(decoded) in the server. The token can be maintained at client side in local storage, session storage or even in cookies.

Third party access, if we have a need to expose our API's outside of our system like third party app or even to access it from mobile apps we end up in two common ways to share the user information.Via **API-token** which is same as JWT token, where the token will be send via Authorization header which will get handled at API gateway to authenticate the user. And the other option is via **Open Authentication(OAuth),**OAuth is a protocol that allows an application to authenticate against server as a user. The recommendation is to implement OAuth 1.0a or OAuth 2.0. OAuth 2.0 relies on HTTPS for security and it currently implemented by Google, Facebook, Twitter etc., OAuth 2 provides secured delegate access to a resource based on user. OAuth 2 does this by allowing a token to be issued by Identity provider to these third party applications, with the approval of user. The client then uses the token to access the resource on behalf of that user.

OpenId is HTTP based protocol that uses identity provider to validate a user. The user password is secured with one identity provider, this allows other service providers a way to achieve Single SignOn(SSO) without requiring password from user. There are many OpenId enabled account on the internet and organizations such as Google, Facebook, Wordpress, Yahoo, PayPal etc., uses OpenId to authenticate users. The latest version of OpenId is OpenId Connect, which provides OpenId(authentication) on top of OAuth 2.0(authorization) for complete security solution.





SAML, Security assertion markup language makes use of the same Identity provider which we saw in OpenId, but it is XML based and more flexible. The recommended version for SAML is 2.0. SAML also provides a way to achieve Single SignOn(SSO), user can make use of the Identity provider URL to login into the system which redirects with XML data back to your application page which can then be decoded to get the user information. We have SAML providers like G Suite, Office 365, OneLogin, Okta etc.,

Which authentication method to pick when?

For Single SignOn OpenId has taken most of the consumer market, SAML is often the choice for many enterprise application.

If you have to support only web application go for Cookie or Token based authentication.

If you have to support both web as well mobile client go with API-token with that of Cookie based authentication.

On top of above authentication methods if needed we can also implement One Time Password(OTP), Two Factor Authentication(2FA), Email verification etc.,

Authentication Jwt Openid Cookies Saml