# How does cookie based authentication work?

Asked 6 years, 8 months ago Active 1 year, 6 months ago Viewed 107k times

191

Can someone give me a step by step description of how cookie based authentication works? I've never done anything involving either authentication or cookies. What does the browser need to do? What does the server need to do? In what order? How do we keep things secure?

I've been reading about different types of authentication and about cookies but I would like a basic description of how to use the two together- I've only read that they are often used together but could not find a description of how.

77

authentication cookies browser

asked Jul 21 '13 at 4:21
Mastid
2,089 ● 2 ● 9 ● 8

3 howstuffworks.com/cookie.htm – Cyclonecode Jul 21 '13 at 4:24

## 3 Answers

Active Oldest Votes

151

A cookie is basically just an item in a dictionary. Each item has a key and a value. For authentication, the key could be something like 'username' and the value would be the username. Each time you make a request to a website, your browser will include the cookies in the request, and the host server will check the cookies. So authentication can be done automatically like that.

To set a cookie, you just have to add it to the response the server sends back after requests. The browser will then add the cookie upon receiving the response.

There are different options you can configure for the cookie server side, like expiration times or encryption. An encrypted cookie is often referred to as a signed cookie. Basically the server encrypts the key and value in the dictionary item, so only the server can make use of the information. So then cookie would be secure

A browser will save the cookies set by the server. In the HTTP header of every request the browser makes to that server, it will add the cookies. It will only add cookies for the domains that set them. Example.com can set a cookie and also add options in the HTTP header for the browsers to send the cookie back to subdomains, like sub.example.com. It would be unacceptable for a browser to ever sends cookies to a different domain.

edited Oct 10 '17 at 12:45
Erik A
24.5k ● 7 ● 30 ● 45

answered Jul 21 '13 at 4:30

Conor Patrick
2,009 ● 4 ● 15 ● 30

What i understand is that the browser are able to send the cookie back to the same domain. In relation to that does browser take subdomain in account when differentiate between two domains? – Aakash Jul 14 '14 at 1:34

1 You can set options in the HTTP header for how a browser handles subdomains. – Conor Patrick Jul 14 '14 at 3:01

266

I realize this is years late, but I thought I could expand on Conor's answer and add a little bit more to the discussion.

> Can someone give me a step by step description of how cookie based authentication works? I've never done anything involving either authentication or cookies. What does the browser need to do? What does the server need to do? In what order? How do we keep things secure?

**Step 1: Client > Signing up**

Before anything else, the user has to sign up. The client posts a HTTP request to the server containing his/her username and password.

**Step 2: Server > Handling sign up**

The server receives this request and hashes the password before storing the username and password in your database. This way, if someone gains access to your database they won't see your users' actual passwords.

**Step 3: Client > User login**

Now your user logs in. He/she provides their username/password and again, this is posted as a HTTP request to the server.

**Step 4: Server > Validating login**

The server looks up the username in the database, hashes the supplied login password, and compares it to the previously hashed password in the database. If it doesn't check out, we may deny them access by sending a 401 status code and ending the request.

**Step 5: Server > Generating access token**

1. Store it in the database associated with that user
2. Attach it to a response cookie to be returned to the client. Be sure to set an expiration date/time to limit the user's session

Henceforth, the cookies will be attached to every request (and response) made between the client and server.

**Step 6: Client > Making page requests**

Back on the client side, we are now logged in. Every time the client makes a request for a page that requires authorization (i.e. they need to be logged in), the server obtains the access token from the cookie and checks it against the one in the database associated with that user. If it checks out, access is granted.

This should get you started. Be sure to clear the cookies upon logout!

edited Aug 31 '15 at 23:10

answered Aug 26 '15 at 4:33

pllx

2,761 ● 1 ● 7 ● 8

8 Thanks for the description. I just wonder how does access token provides security? Can an attacker if steals the cookie, pose as an authenticated logged in user? Or that is protected by SSL? – Richeek Oct 3 '15 at 19:45

6 @Richeek SSL secures interception during requests/responses, but an attacker might access your cookies at the endpoints (e.g. your browser). Theoretically, they could then pose as a logged in user until the cookie expires. I say “theoretically” because the implementation above doesn’t handle that. In the above implementation, the attacker will have access until the access token in your database is updated (i.e. next login). – pllx Oct 7 '15 at 3:17

14 You might invalidate the access token upon expiry yourself, perhaps with an “expiration date” in your database. Or, you could consider using JSON Web Tokens (JWT), which are like access tokens, but can handle token expiry among other things. More on JWT here. An attacker will still have access to your account for brief periods of time if they have your access token/JWT, so you should also secure your endpoints. – pllx Oct 7 '15 at 3:17

3 Took me long to say thank you! Thanks for your explanantion – Richeek Feb 10 '16 at 23:20

2 @ManuChadha you could along with the token/session key also save the user's ip address along with other identifying parameters such as user-agent, etc. if the request then comes with a valid cookie but from the wrong ip, browser, etc then you deny the request and redirect the user to the login page to authenticate again. – FalcoGer Dec 11 '19 at 13:11

16

**Cookie-Based Authentication**

Cookies based Authentication works normally in these 4 steps-

1. The user provides a username and password in the login form and clicks Log In.
2. After the request is made, the server validate the user on the backend by querying in the database. If the request is valid, it will create a session by using the user information fetched from the database and store them, for each session a unique id called session Id is created

3. Browser will submit this session Id on each subsequent requests, the session ID is verified against the database, based on this session id website will identify the session belonging to which client and then give access the request.
4. Once a user logs out of the app, the session is destroyed both client-side and server-side.

answered Feb 10 '18 at 5:45

Debendra Dash
**3,210** ● 24 ● 28

**Highly active question**. Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.