Theory: HTTP Basic Auth

10 minutes 0 / 5 problems solved

Websites often require login and password to sign in. We've all had to enter this data, whether it is mail, some social network or a forum. After successfully entering login and password, a number of additional features become available, for example, full access to content, ability to correspond and leave comments.

The process of issuing rights to perform certain actions is called user **authorization**.

§1. Benefits of authorization

Yes, authorization allows you to identify a visitor of your web page; however, at the same time it limits access rights for unauthorized users. Hence you might have some doubts: why put restrictions at all, isn't it easier to open full functionality of the site to all visitors? After all, it's a fact that people don't like unnecessary time-consuming procedures, even if registration and authorization take only a couple of minutes.

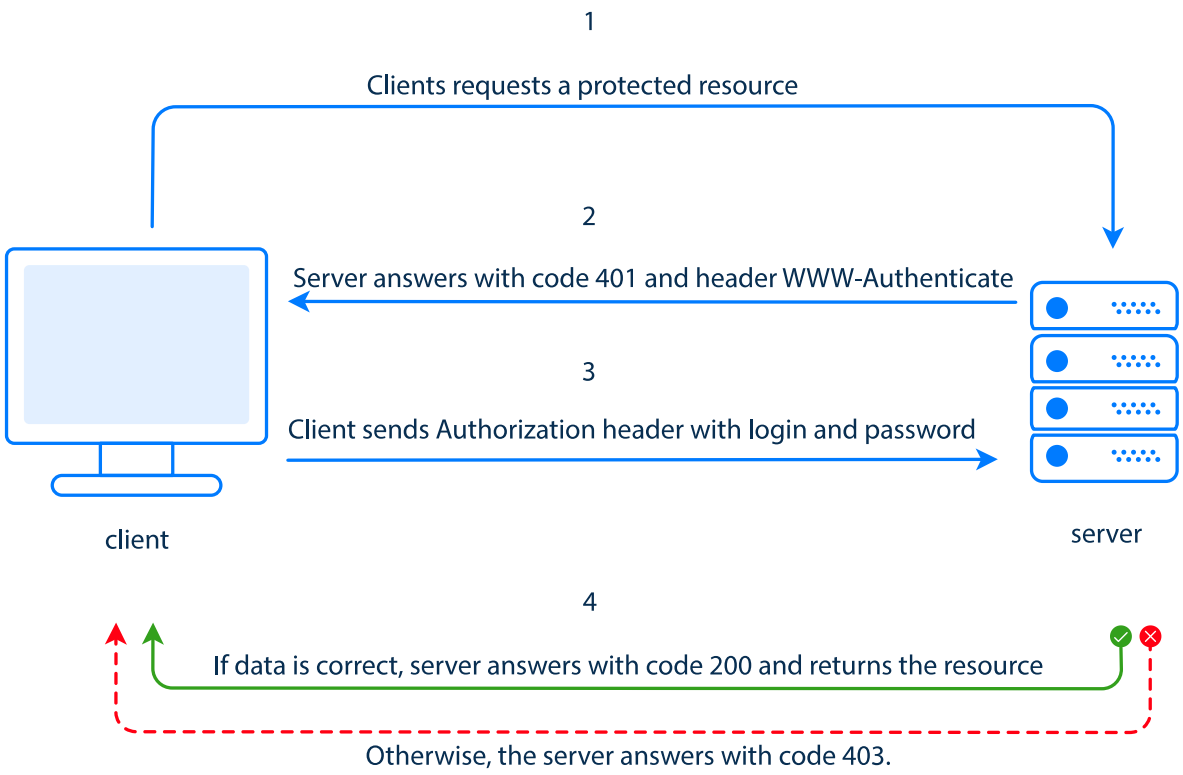
Well, authorization actually has many benefits. It lets you flexibly manage your personal data, allows commercial websites to offer additional services for a fee and better protects your confidential information. Increased security is perhaps the greatest advantage of authorization.

§2. Basic authorization in HTTP

HTTP has a built-in mechanism for authorization. The easiest HTTP authorization scheme is **"Basic"**. It relies on login and password. Let's see how it works.

1. When a user enters a URL in the browser's address bar, they send a request to access the desired resource.
2. If the resource is protected, the server requires authorization from the user. It responds to the client with the HTTP status code **401 (Unauthorized)** and the header `WWW-Authenticate`. When the browser receives this code, it shows a pop-up window where the user must enter their login and password.
3. The user enters them, and the browser repeats the request to the same resource. Transmitting authorization data to the server is performed using the `Authorization` header, in which the encoded login and password are written.
4. After the web server receives the request with the specified header, it checks that login and password are correct. If both are entered correctly, the web server grants access to the resource. The response code is **200 (OK)**. If the data is incorrect, the response code is **403 (Forbidden)**, and the user will be informed about an error in the entry and that access to the desired information is denied.

The following picture shows the sequence of the authorization algorithm:



§3. Creating an HTTP header

Usually, adding a header with login and password to the request is done by the browser, but if you need to create your own library to work with HTTP for software compatibility with web services, you need to understand how this works under the hood.

The header syntax for basic authorization looks as follows:

```
Authorization: <type> <credentials>
```

`<type>` denotes the type of authorization. In this case, we are looking at the `Basic` type.

If the `Basic` authentication scheme is used, the `<credentials>` are constructed like this:

- the username and the password are combined with a colon (`student:ilovetostudy`);
- the resulting string is base64 encoded (`c3R1ZGVudDppbG92ZXRvc3R1ZHk=`).

Here is a valid example of an authorization header:

```
Authorization: Basic c3R1ZGVudDppbG92ZXRvc3R1ZHk=
```

> The **base64** encoding is not a secure representation of credentials since it does not mean encryption or hashing. An encoded string can be easily decoded into the original form.

There are sites that can help you generate the `Authorization` header for your credentials.

§4. Security and basic auth

In the **Basic auth** scheme, a client must send login and password every time they try to access a protected resource. Sending a non-encrypted password can be too dangerous (base64 is not an encryption), so it is better to use secure HTTP (HTTPS) with it.

There are more convenient and secure authorization methods which help to avoid the frequent sending of login and password. Understanding the principles of HTTP authorization will help you quickly understand the essence of working with more complex schemes of authorization.

Report a typo

115 users liked this theory. **1** didn't like it. **What about you?**

Start practicing

Comments (3) Hints (0) Useful links (1) Show discussion