Twitter OAuth example using cURL

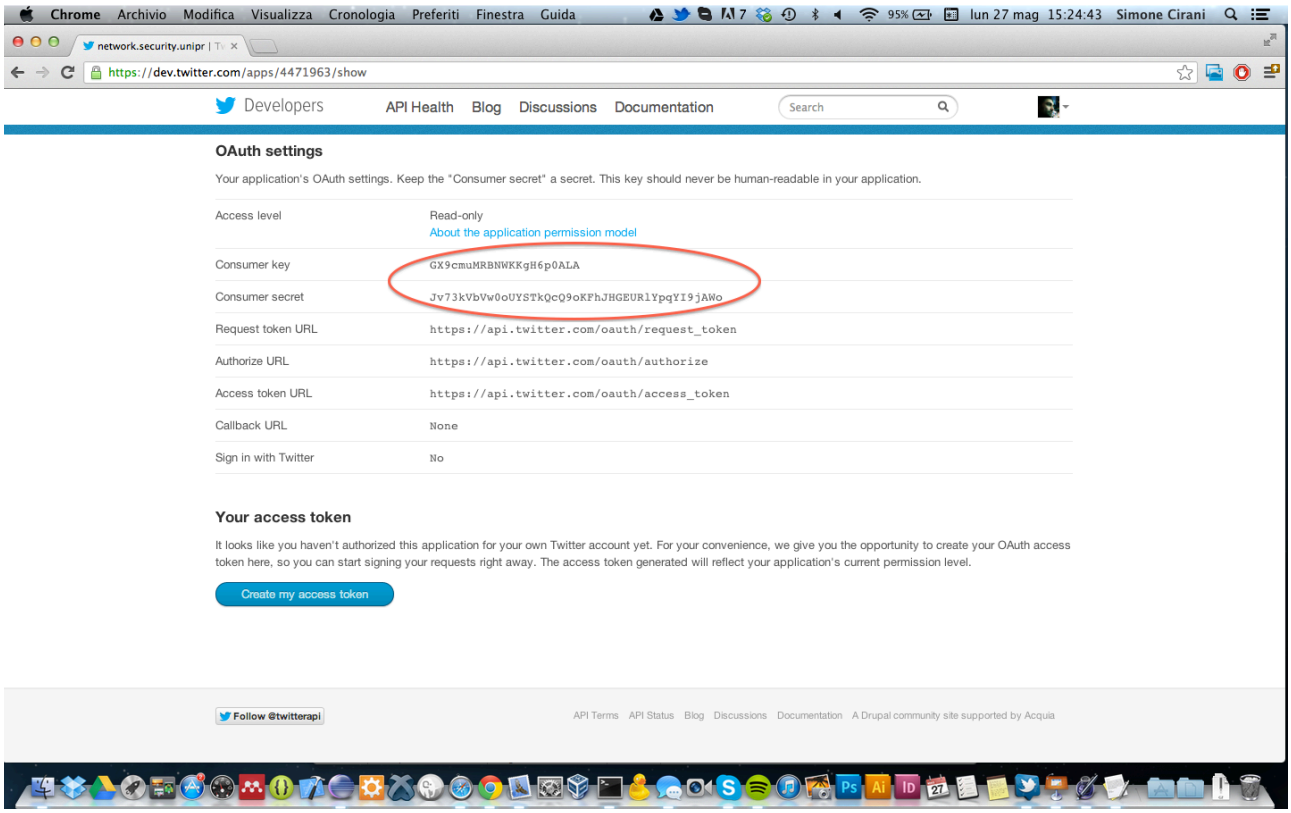
This exercise will walk you through all the steps involved in performing an authenticated request to Twitter from a custom client application.

The first thing we need to do is to create a new client application, get the consumer keys and secret and access token and secret.

1. Log in to the Twitter Developers website (https://dev.twitter.com/apps) and click on the “Create a new application” button.

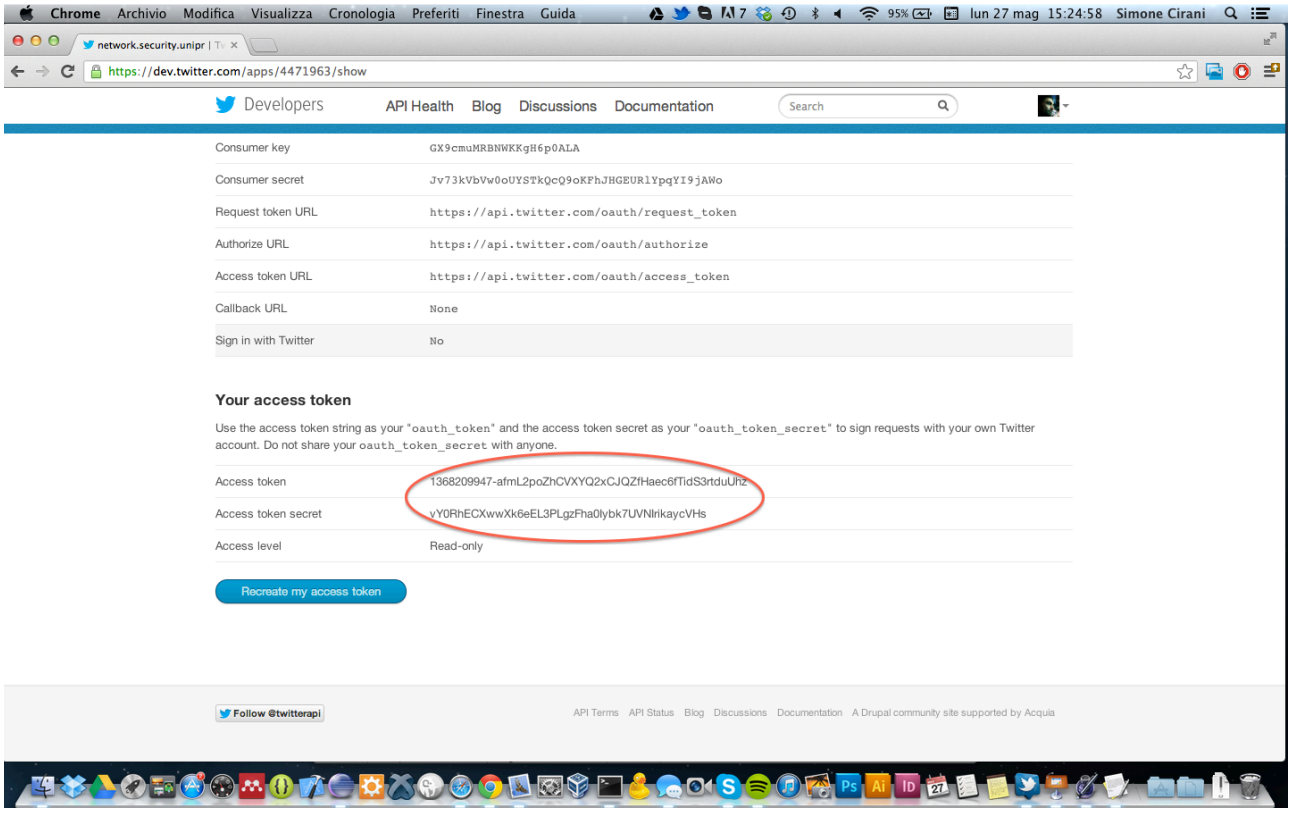
2. Enter the name of the client application, its description, and its website URL (this is where the user can retrieve information about the application).
   Next, agree Twitter’s Developers Rules, fill the captcha, and then submit.

3. Now the application has been created. You will be presented the application’s details page. In this page, you will be able to retrieve the consumer’s key and secret. These information must be kept secret or other parties might act as if they were your application.
   Note that the details will show also the access level, that is what type of actions the application will be able to perform. In this case read-only level is fine, since we are going to get the user timeline. If we were to write tweets, we should have changed this “Read and Write” or “Read, Write and Access direct messages”. More details on Application Permission Model can be found at https://dev.twitter.com/docs/application-permission-model.



Next, click on the “Create my access token” button at the bottom of the page. This will perform a 2-legged authentication, which will link your account to the client application. This means that you will not be asked to perform a login to grant permission to your application to access your data.

4. Reload the page and you will be able to get the generated access token and secret. As for the consumer key and secret, these must be kept secretly or someone else could be using them in order to get your personal data.



5. Now you have your consumer key and secret and access token and secret pairs which we can use to perform authenticated requests.

6. In order to get the user timeline we will use the user timeline service, whose Request URL is:

https://api.twitter.com/1/statuses/user_timeline.json?screen_name=<SCREEN_NAME>

where the query screen_name=<SCREEN_NAME> refers to the user we are targeting.

Now, we are going to write a short script that will construct a valid HTTP request that will retrieve the user timeline.

7. First, we set the consumer key and secret and access token and secret pairs in variables:

consumer_key=<YOUR CONSUMER KEY>

```
consumer_secret=<YOUR CONSUMER SECRET>
access_token=<YOUR ACCESS TOKEN>
access_token_secret=<YOUR ACCESS TOKEN SECRET>
```

1. Next, we set the screen name of the user we are targeting. Since we are going to use the access token and secret of a 2-legged aithentication, just set it to your account screen name:

```
screen_name=<YOUR SCREEN_NAME>
```

2. Next, we are going to get the current UNIX timestamp (seconds since Jan. 1st, 1970 at 00:00:00) with the command:

```
timestamp=`date +%s`
```

3. Now, we can create the nonce. Since the nonce must be used only once, we can use the following procedure:
   a) take the current timestamp
   b) Base64-encode it
   c) replace all '+', '/', and '=' signs with %hexcode

```
nonce=`date +%s%T%N | openssl base64 | sed -e s'/[+=/]//g'`
```

4. Now we are ready to create the Base Signature String, using the rules defined by the OAuth protocol, and sign it using the HMAC-SHA-1 scheme:

```
signature=`echo -n 'GET&https%3A%2F%2Fapi.twitter.com%2F1%2Fstatuses
%2Fuser_timeline.json&oauth_consumer_key%3D'$consumer_key'%26oauth_nonce
%3D'$nonce'%26oauth_signature_method%3DHMAC-SHA1%26oauth_timestamp
%3D'$timestamp'%26oauth_token%3D'$access_token'%26oauth_version
%3D1.0%26screen_name%3D'$screen_name | openssl dgst -sha1 -hmac
$consumer_secret'&'$access_token_secret'' -binary | openssl base64 | sed -e s'/+/%2B/' -
e s'/\//%2F/' -e s'/=/%3D/'`
```

5. Finally, we can do the actual request by using the cURL client:

```
curl --get 'https://api.twitter.com/1/statuses/user_timeline.json' --data "screen_name=
$screen_name" --header 'Authorization: OAuth oauth_consumer_key="'$consumer_key'",
oauth_nonce="'$nonce'", oauth_signature="'$signature'",
oauth_signature_method="HMAC-SHA1", oauth_timestamp="'$timestamp'",
oauth_token="'$access_token'", oauth_version="1.0"' --verbose
```

If everything has been done correctly, the request will return a **HTTP/1.1 200 OK** whose body will contain your timeline in JSON format.
If the consumer key/secret, or access token/secret, or the signature were not correct, you will receive a **HTTP/1.1 401 Unauthorized** response.
If the request was malformed, you will receive a **HTTP/1.1 400 Bad Request** response.